12

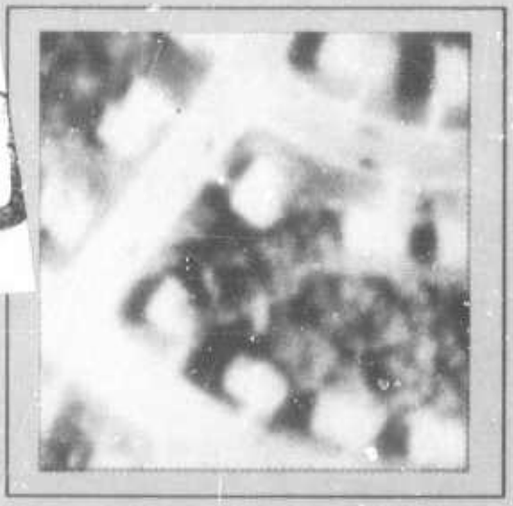
PROCEEDINGS:

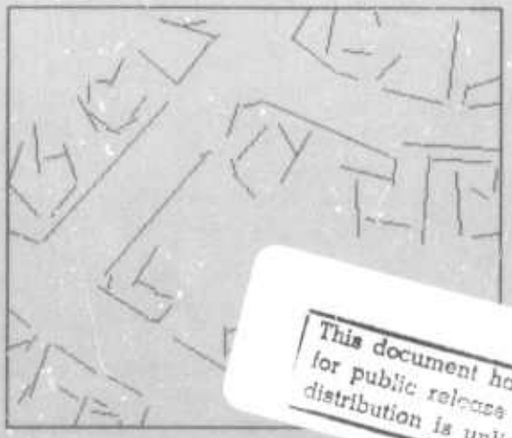
IMAGE UNDERSTANDING WORKSHOP

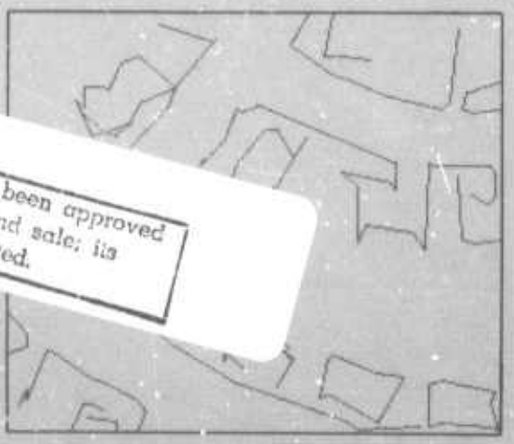
APRIL 1980

Sponsored by:
Information Processing Techniques Office
Defense Advanced Research Projects Agency

Hosted by:
University of Maryland
Computer Vision Laboratory







STEPS IN THE RECOGNITION
OF CULTURAL FEATURES

SCIENCE APPLICATIONS, INC.

Science Applications, Inc.

(6)

IMAGE UNDERSTANDING.

Proceedings of a Workshop (11)
Held at
College Park, Maryland,
April 30, 1980.

Sponsored by the
Defense Advanced Research Projects Agency

(9) Semi-annual technical rept. Nov 79-Apr 80, (12) 218

(14)
Science Applications, Inc.
Report Number SAI-81-170-WA

(15) MDA903-80-C-0188, DARPA Order-3456

(10) Lee S. Baumann
Workshop Organizer and
Proceedings Editor

(11) Apr 80

407154

This report was supported by
The Defense Advanced Research
Projects Agency under DARPA
Order No. 3456, Contract No. MDA903-80-C-0188
Monitored by the
Defense Supply Service, Washington, D.C.

407154 slk

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: SAI-81-170-WA	2. GOVT ACCESSION NO.: AD-A084 764
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): IMAGE UNDERSTANDING Proceedings of a Workshop, April, 1980	5. TYPE OF REPORT & PERIOD COVERED: SEMI ANNUAL TECHNICAL NOVEMBER, 1979-APRIL, 1980
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): LEE S. BAUMANN (Ed.)	8. CONTRACT OR GRANT NUMBER(s): MDA903-80-C-0188 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS: SCIENCE APPLICATIONS, INC. 1911 North Fort Myer Drive, Suite 1200 Arlington, Virginia 22209	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA ORDER No. 3456
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209	12. REPORT DATE: APRIL, 1980 13. NUMBER OF PAGES: 209
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing; Image Understanding; Scene Analysis; Edge Detection; Image Segmentation; CCDArrays; CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the technical papers and outlines of semi-annual progress reports presented by the research activities in Image Understanding, sponsored by the Information Processing Techniques Office; Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 30 April 1980 in College Park, Maryland.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

TABLE OF CONTENTS

PAGE

FORWARD i

AUTHOR INDEX. iii

SESSION I - PROGRAM REVIEWS BY PRINCIPAL INVESTIGATORS

"Image Understanding Using Overlays"
A. Rosenfeld; University of Maryland 1

"Image Understanding Research at CMU: A Progress Report"
J.R. Kender and R. Reddy; Carnegie-Mellon University 13

"Understanding Images at MIT: Representative Progress"
K. Stevens, K. Nishihara, B. Schunck, the Staff; Massachusetts Institute of Technology . . 15

"Progress at the Rochester Image Understanding Project"
J.A. Feldman, K.R. Sloan, Jr.; The University of Rochester 20

"Spatial Understanding"
T.O. Binford; Stanford University. 24

"The SRI Image Understanding Program"
M.A. Fischler; SRI International 28

"Progress in Image Understanding Research at USC"
R. Nevatia, A.A. Sawchuk; University of Southern California. 31

SESSION II - TECHNICAL PAPERS

"Toward the Recognition of Cultural Features"
M. Tavakoli; University of Maryland. 33

"Atmospheric Modelling for the Generation of Albedo Images"
R.W. Sjoberg, B.K.P. Horn; MIT Artificial Intelligence Laboratory. 58

"Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"
M.A. Fischler, R.C. Bolles; SRI International. 71

"Semantic Description of Aerial Images Using Stochastic Labeling"
O.D. Faugeras, K.E. Price; University of Southern California 89

"Representing and Reasoning about Partially Specified Scenes"
R.A. Brooks, T.O. Binford; Stanford University 95

TABLE OF CONTENTS (CONT.)

SESSION II - TECHNICAL PAPERS (Cont.)

PAGE

"A Storage Representation for Efficient Access to Large Multi-Dimensional Arrays"
L.H. Quam; SRI International 104

"Some Uses of Pyramids in Image Processing and Segmentation"
A. Rosenfeld; University of Maryland 112

"Solving for the Parameters of Object Models from Image Descriptions"
D.G. Lowe; Stanford University 121

"Aspects of a Computational Theory of Human Stereo Vision"
W.E.L. Grimson; MIT Artificial Intelligence Laboratory 128

"Experience with the Generalized Hough Transform"
K.R. Sloan, Jr., D.H. Ballard; The University of Rochester 150

"The Gaussian Sphere: A Unifying Representation of Surface Orientation"
J. Kender; Carnegie-Mellon University. 157

"Object Detection and Measurement Using Stereo Vision"
D.B. Gennery; Stanford University. 161

"Edge-Based Stereo Correla ion"
H. Baker; Stanford University. 168

"Convergence Properties of Two Label Relaxation"
A. Helland; Westinghouse Systems Development Division. 176

"Investigation of VLSI Technologies for Image Processing"
W.L. Eversole, D.J. Mayer; Texas Instruments Incorporated. 182

"Application of LSI and VLSI to Image Understanding Architectures"
S.D. Fouse, G.R. Nudd, V.S. Wong; Hughes Research Laboratories 190

"A 'Non-Correlation' Approach to Image-Based Velocity Determination"
O. Firschein; Lockheed Palo Alto Research Laboratory 195

"Bootstrap Stereo"
M.J. Hannah; Lockheed Palo Alto Research Laboratory. 201

"Status Report on the Advanced Flexible Processor"
G.R. Allen; Control Data Corporation 209

FORWARD

The Eleventh Image Understanding Workshop sponsored by the Defense Advanced Research Projects Agency (DARPA), Information Processing Techniques Office was held in College Park, Maryland on April 30, 1980. The Program Manager, Lt. Col. Larry E. Druffel of DARPA opened the streamlined one day program. Research personnel from the various university and industrial firms active in the program and officials from Army, Navy, Air Force and other government agency laboratories and research facilities were in attendance. Lt. Col. Druffel advised the more than one hundred participants at the workshop that the Image Understanding Program sponsored by the Defense Advanced Research Projects Agency is continuing to mature and new techniques continue to advance our ability to eventually derive information from images. The cartographic testbed, Druffel noted, is well into the planning stages and initial systems software is presently under development.

Session I of these proceedings contains outlines of the program progress reports presented by the seven principal investigators during the morning session. In order to facilitate the interchange of ideas between the research community and the government attendees which was one of the principle goals for the workshop, the afternoon session opened with a discussion of the role of scene models in image processing. Several expert speakers discussed various aspects of that subject. The remainder of the afternoon session was devoted to the presentation of significant technical papers which were selected from those prepared for this workshop. In the Session II section of these proceedings all papers submitted are reproduced in order that the contents be available to those interested persons receiving copies of these proceedings, although only a few were actually reviewed at the workshop.

In response to lower travel budgets, the formal program for this workshop was limited to one full day rather than the two days usually allotted. In anticipation of continued austerity in travel, DARPA announced that future workshops will be held annually instead of semi-annually. In the belief that the interchange of ideas is important to the program, future annual workshops will consist of two day sessions conducted each spring.

This workshop was hosted by Dr. Azriel Rosenfeld, head of the Computer Vision Laboratory at the University of Maryland. The Workshop Organizer wishes to express his appreciation to Dr. Rosenfeld for his invaluable assistance in arranging for this workshop and for his efforts in selecting and organizing the program. Appreciation is also due Ms. Ramona C. Hall and Ms. Nancy A. Smith of Science Applications, Incorporated for their assistance for putting together these proceedings while under difficult deadlines.

The materials for the cover of this document were selected by Dr. Rosenfeld as representative of some of the work underway in his laboratory. The picture, stated Dr. Rosenfield, is an image of part of a suburb in the Occoquan, Virgina area. The three diagrams are respectively: line segments fitted to edge points in the picture, segments that could be pieces of roads or buildings based on the adjacent average gray levels, and the results of linking compatible segments. More detail on these pictures is contained in the included paper entitled, "Toward the Recognition of Cultural Features," by Dr. M. Tavakoli. The art work and layout for the cover was created by Miss Elody Blomberg of the Art Department of Science Applications, Incorporated.

Lee S. Baumann
Science Applications, Inc.
Workshop Organizer

AUTHOR INDEX

NAME	PAGE
G.R. Allen	209
H. Baker	168
D.H. Ballard	150
T.O. Binford	24, 95
R.C. Bolles	71
R.A. Brooks	95
W.L. Eversole	182
O.D. Faugeras	89
J.A. Feldman	20
O. Firschein	195
M.A. Fischler	28, 71
S.D. Fouse	190
D.B. Gennery	161
W.E.L. Grimson	128
M.J. Hannah	201
A. Helland	176
B.K.P. Horn	58
J.R. Kender	13, 157

AUTHOR INDEX (Cont.)

NAME	PAGE
D.G. Lowe	121
D.J. Mayer	182
R. Nevatia	31
K. Nishihara	15
G.R. Nudd	190
K.E. Price	89
L.H. Quam	104
R. Reddy	13
A. Rosenfeld	1, 112
A.A. Sawchuk	31
B. Schunck	15
R.W. Sjoberg	58
K.R. Sloan, Jr.	20, 150
K. Stevens	15
M. Tavakoli	33
V.S. Wong	190

SESSION I

PROGRAM REVIEWS

BY

PRINCIPAL INVESTIGATORS

IMAGE UNDERSTANDING USING OVERLAYS
PROJECT STATUS REPORT, 1 OCTOBER 1979-30 MARCH 1980
CONTRACT DAAG-53-76C-0138 (DARPA ORDER 3206)

Azriel Rosenfeld
Principal Investigator

Computer Vision Laboratory
Computer Science Center, University of Maryland
College Park, MD 20742

ABSTRACT

Current activities on the project are reviewed under the following headings:

1. Segmentation and texture analysis
2. Local and global shape analysis
3. Hierarchical representation

INTRODUCTION

This project is concerned with the study of advanced techniques for the analysis of reconnaissance imagery. It is being conducted under Contract DAAG-53-76C-0138 (DARPA Order 3206), monitored by the U.S. Army Night Vision Laboratory, Ft. Belvoir, VA (Dr. George Jones). The Westinghouse Systems Development Division, as a subcontractor, is investigating implementation of the techniques being developed by Maryland, particularly in the area of relaxation; their efforts are reviewed in separate quarterly reports.

The current phase of the project is concerned with the development and application of advanced techniques for image processing, feature detection, segmentation, texture and shape analysis, and region representation. These aspects are reviewed in the following sections. This report deals primarily with the work done during the past six months; activities during earlier periods were reviewed in previous reports [1-6]. Some of the topics are discussed only briefly, since they are treated in greater detail in individual technical reports and Image Understanding Workshop papers.

SEGMENTATION AND TEXTURE ANALYSIS

2.1 Edge detection

Edges are generally detected by thresholding the output of some type of difference operator; but the choice of a threshold for this purpose is not easy, since the histogram of difference values tends to fall off smoothly from a peak near zero. Threshold selection becomes easier if we suppress nonmaximum difference values (in the gradient direction) before histogramming. As Figure 1 shows, this yields a histogram composed of a sharpened peak near zero together with small sets of higher values; the latter are likely to be good choices for edge points [7].

Difference operators for edge detection can be designed by fitting a polynomial surface to the gray levels in the neighborhood of a point, and taking the gradient of that surface as an estimate of the image gradient. This approach can be generalized to the design of operators for surface detection in three- (or higher-) dimensional arrays of data, such as those obtained by reconstructing objects from x-ray projections, by stacking cross-sections, or by stacking successive frames in a time sequence of images. Surface detection provides results that are more accurate and more reliable than those obtained by applying two-dimensional edge detectors to the individual slices, as can be seen from Figure 2 [8]. This work is part of a Ph.D. thesis on processing and segmentation of three-dimensional arrays.

2.2 Pixel classification and texture analysis

During the past reporting period, an M.S. thesis was completed [9] on a general-purpose software package for performing relaxation operations on arrays of pixels. This package allows the user to specify the process for computing initial probabilities, the neighborhood to be used, and the probability adjustment algorithm (including the compatibility coefficients). As an application, a light/dark relaxation process was implemented; examples of this process can be found in earlier status reports [2,5]. Some analytical results regarding such two-label relaxation processes will be presented in a forthcoming quarterly report on the Westinghouse subcontract.

Relaxation has been successfully used to improve pixel classification based on color, as reported elsewhere. It can similarly be used to improve pixel classification in single-band images based on local property values such as gray level and "busyness". Figure 3a shows a house picture containing five principal types of regions--sky, grass, bushes, brick, and shadow. The bush and shadow classes are very difficult to distinguish; they have similar mean vectors, and the bush class is more variable, so that a maximum-likelihood classification (based on Gaussian fitting to the clusters defined by hand segmentation) misclassifies most of the shadow pixels as bush (Figure 3b). The results are greatly improved when relaxation is used to adjust the initial class probabilities for each pixel based on those of its neighbors; see Figure 3c. Similar improvement is obtained

when the busyness values are iteratively smoothed, e.g. by median filtering, prior to clustering and classification (Figure 3d). Further details on these experiments can be found in [10].

Iterative smoothing can also be used tn improve the results of texture classification using texture features derived from small windows, as described in [11-12].

2.3 Interactive segmentation

An interactive image segmentation system is being designed as a contribution to the DARPA/DMA Testbed. The system allows the user to designate samples of two classes (e.g., objects and background). It analyzes the samples, designs a classifier to discriminate them, and displays the classification results to the user for evaluation; if errors are designated, the system attempts to modify the classifier so as to eliminate them. The user need not know anything about the classification process or the features that are used for classification; the system selects them from a prespecified repertoire. The current, pilot version of the system classifies pixels based on gray level only; future versions will make use of various types of local features and will allow more than two classes.

LOCAL AND GLOBAL SHAPE ANALYSIS

3.1 Corner detection

Several types of operators have been developed that respond to the presence of "corners" (i.e., sharp changes in edge direction) in an unsegmented image [13]. For example, one can express the rate of change in the gradient direction in terms of first and second derivative operators, and then approximate these by difference operators; or one can simply compute a digital gradient direction, and estimate its rate of change at P by comparing it with the directions at the appropriate neighbors of P. To measure "cornerity", the rate of change in gradient direction should be multiplied by the gradient magnitude, since we are only interested in corners that lie on edges. Figure 4 shows a display of cornerity values for a simple grayscale image; the results seem reasonable.

3.2 Collinearity and parallelism

Collinear and parallel (or "antiparallel") sets of edge and line segments are important elements in the description of many types of scenes. The following paragraphs describe general-purpose programs for analyzing collinearity and parallelism. A more specialized program that links edge segments based on gray level, as well as geometric, criteria, with application to the detection of buildings and roads on aerial imagery, is described in a separate paper in these proceedings [14].

The "collinearity strength" of two segments depends on several factors:

(a) The distance between their nearer ends, relative to their lengths
(b) The angles that they make with the line joining their nearer ends
(c) The distance between their farther ends, relative to the nearer-end distance and lengths.

A collinearity strength measure based on a combination of these factors gives generally reasonable results, as illustrated in Figure 5 [15].

Collinear segments can be grouped into "clusters" based on their relative sizes and separations. Several types of cluster merit functions can be used for this purpose; a good figure of merit should depend on both the segment density and the total segment length in the given cluster. Examples of clusters defined by maximizing such a figure of merit are given in Figure 6. A report on these experiments is in preparation [16].

Segments can also be linked based on parallelism (or, in the case of edge segments, antiparallelism: the dark sides of the edges should face in opposite directions). The figure of merit for this linking process should depend on the separation of the segments, their lengths and the amount by which they overlap, as well as their parallelism (i.e., the angle between them). Mutually best pairs based on this merit function can be linked, and the process can then be repeated with the linked pairs eliminated. For examples of the results obtained using this approach see [17].

3.3 The medial axis

The medial axis (MA) of a set S is defined as the set of centers (and radii) of the maximal "disks" contained in S, or equivalently, as the set of points of S whose distances from the complement $\bar{S}$ are local maxima. It can be used as a compact representation of S, and can also serve as a basis for approximating S by a union of "generalized ribbons" (= connected arcs of MA points, with radii specified by a "width function" defined along each arc).

The MA is sensitive to noise, i.e., to errors in extracting the set S; thus it would be desirable to define it directly for unsegmented images. This can be done using a "gray-weighted" concept of distance, but it is hard to reconstruct the image from such an MA. Another possibility (the "SPAN": Spatial Piecewise Approximation by Neighborhoods) is to approximate the image by maximal homogeneous disks, but the approximation process is computationally costly. Still another alternative is to assign an MA score to each point P based on the presence of high gradient values at pairs of positions symmetrically located with respect to P; but this process turns out to be quite sensitive to noise.

A more robust approach to defining an MA for unsegmented images is based on a characterization of the MA of a set S in terms of shrinking and expanding operations performed on S. Let $S^{(-n)}$ denote the result of shrinking S (i.e., deleting

its border) n times, and similarly let $S^{(n)}$ denote the result of expanding S n times ($S^{(n)} = \overline{S^{(-n)}}$). Then it is not hard to see that $S_k \equiv (S^{(-k)})(1) - \overline{S}^{(-k+1)}$ is the set of MA points at distance k from S, so that $\cup S_k$ is the MA. To generalize this to unsegmented images, we use local MIN operations instead of shrinking, and local MAX operations instead of expanding; we can then define the "MMMAT" (= min-max MAT) as ΣS_k. Examples of such MMMATs are shown in Figure 7; for further details see [18]. Approximations to the image can be reconstructed by using only points having strong MMMAT values, and k's that make strong contributions to these values. For examples of such reconstructions see Figure 8; a report on this work is in preparation.

3.4 Shape segmentation

Various types of shape features, such as protrusions and intrusions, can be detected by comparing boundary arcs with their chords; for example, if the chord is much shorter than the arc, or if the arc does not lie close to the chord, that arc must be a protrusion or intrusion. Suppose that we measure various arc-chord figures of merit (e.g., arc length divided by chord length, or area between arc and chord divided by squared chord length) for every arc. In many cases, extrema of such figures of merit correspond to arcs that are natural "pieces" of the shape, as illustrated in Figure 9. However, this approach sometimes leads to segmentations that are not intuitively plausible, since the extrema depend only on (e.g.) the curves slopes at the arc endpoints, and not on the shape of the arc between the endpoints; see [19].

Work on shape segmentation using relaxation, described in earlier reports, is being extended to handle shapes with major occlusions or missing parts; the results will be described in a forthcoming report.

HIERARCHICAL REPRESENTATION

4.1 Quadtrees and nextrees

The quadtree algorithms developed on this project usually involve locating neighbors of a given block in the image by searching the tree starting from the corresponding node. A general treatment of neighbor finding in quadtrees, including an analyis of the expected computational costs, can be found in [20].

Quadtrees are defined on the basis of recursive subdivision into quadrants; they involve square blocks, and four blocks of a given size constitute a block of the next larger size. For some purposes it may be desirable to define a representation based on hexagonal rather than square blocks, since such a representation would be less sensitive to rotation. Hexagons cannot be combined to form exact hexagons of a larger size, but one can combine seven hexagons into a "ragged" hexagon, and this process can be iterated, as illustrated in Figure 10. A detailed discussion of how to define hexagonal "pyramids" in this way can be found in [21].

4.2 Quadtree shape approximation

When a region is represented by a quadtree, the upper levels of the tree, corresponding to large blocks of the image, define approximations to the region. These approximations can be used to estimate shape properties such as moments, and to speed up shape matching by eliminating gross mismatches rapidly [22]. For example, the coordinates of the centroid of a shape can be estimated to a fraction of a pixel using quadtree approximations, as illustrated in Figure 11. This should make it possible to track moving shapes quite accurately; even theough the quadtree itself changes radically when a shape is shifted, the moment approximations remain stable. Similarly, the approximations can be used to determine upper and lower bounds on the mismatch area; thus if we are matching a given shape S_j against a collection of stored shapes $S_1, S_2, \ldots$, we can eliminate any S_i such that the lower bound on the mismatch of S_j with S_i exceeds the upper bound on the mismatch of S_j with some other shape. This error bounding process is illustrated in Figure 12.

4.3 Hierarchical image processing and segmentation

Extensive work is now in progress on the use of pyramid structures for image processing and segmentation. This work is summarized in a separate paper in these Proceedings [23]. The following are some of the chief areas of investigation:

a) Iterated local convolution operations can be used to produce large-kernel convolutions having almost exactly Gaussian kernels. These can in turn be combined to yield various types of circular or elongated center-surround operators [24].

b) Image pyramids can be defined in which the blocks at each level overlap; this largely negates the objections to conventional power-of-2 pyramids on grounds of shift sensitivity.

c) In an overlapped pyramid, by associating nodes with their most similar ancestors, one can establish linked clusters of nodes representing homogeneous regions; this facilitates smoothing or segmentation of the regions.

d) Local operations in a pyramid can be used to detect simple types of objects in the image, and to extract these objects by local thresholding. This approach was applied to blob-like objects in an earlier report; it has now been extended to streak-like objects [25].

e) Pyramids can be used to define quadtree approximations to an image ("Q-images"), based on the concept that a block is subdivided only if it is unhomogeneous.

f) The use of Q-images facilitates segmentation by thresholding, since the peaks in the histogram of a Q-image (where each block contributes its mean gray level, a number of times proportional to its size) tend to be sharper and more cleanly separated. The histogram is further improved when we eliminate small blocks, since these tend to lie on region borders. Conversely, if we histogram only the small blocks, we obtain a unimodal histogram whose mean is a good threshold [26]. More generally, we can find blocks in the quadtree corresponding to peaks in the histogram, and apply local thresholds in the vicinity of these blocks to extract the appropriate regions [27].

g) Q-images can also be used to improve edge detection, based on establishing correspondences between edges in the Q-image and edges in the original image [28].

REFERENCES

1. Semi-annual report, 1 April-30 September 1978.
2. Semi-annual report, 1 October 1978-30 March 1979.
3. Semi-annual report, 1 April-30 September 1979.
4. Project status report, in Proceedings, Image Understanding Workshop, November 1978, 20-27.
5. Project status report, in Proceedings, Image Understanding Workshop, April 1979, 14-24.
6. Project status report, in Proceedings, Image Understanding Workshop, November 1979, 166-175.
7. L. Kitchen, Non-maximum suppression of gradient magnitudes makes them easier to threshold, in L. Kitchen, A. Broder, and A. Rosenfeld, Two notes on digital edges and lines, TR-885, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1980.
8. D. G. Morgenthaler and A. Rosenfeld, Multidimensional edge detection by hypersurface fitting, TR-877, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, February 1980.
9. R. C. Smith, A general-purpose software package for array relaxation, TR-839, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1979.
10. P. A. Dondes and A. Rosenfeld, Pixel classification based on gray level and local "busyness", TR-874, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1980.
11. A. Rosenfeld, Cooperative computation in texture analysis, in Proceedings, Image Understanding Workshop, November 1979, 52-56.
12. T. H. Hong, A. Y. Wu, and A. Rosenfeld, Feature value smoothing as an aid in texture analysis, TR-844, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1979.
13. L. Kitchen and A. Rosenfeld, Gray level corner detection, TR-887, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1980.
14. M. Tavakoli, Toward the recognition of cultural features, in Proceedings, Image Understanding Workshop, April 1980.
15. A. Broder and A. Rosenfeld, A note on collinearity merit, in L. Kitchen, A. Broder, and A. Rosenfeld, Two notes of digital edges and lines, TR-885, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1980.
16. A. Scher, M. Shneier, and A. Rosenfeld, Clustering of collinear line segments, TR-888, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, April 1980.
17. A. Scher, M. Shneier, and A. Rosenfeld, A method for finding pairs of anti-parallel straight lines, TR-845, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1979.
18. S. Peleg and A. Rosenfeld, A min-max medial axis transformation, TR-856, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1980.
19. W. S. Rutkowski, Shape segmentation using arc/chord properties, TR-849, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1979.
20. H. Samet, Neighbor finding techniques for images represented by quadtrees, TR-857, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1980.
21. P. J. Burt, Tree and pyramid structures for coding hexagonally sampled binary images, TR-814, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, October 1979.
22. S. Ranade, A. Rosenfeld, and H. Samet, Shape approximation using quadtrees, TR-847, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1979.
23. A. Rosenfeld, Some uses of pyramids in image processing and segmentation, Proceedings, Image Understanding Workshop, April 1980.
24. P. J. Burt, Fast, hierarchical correlations with Gaussian-like kernels, TR-860, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1980.

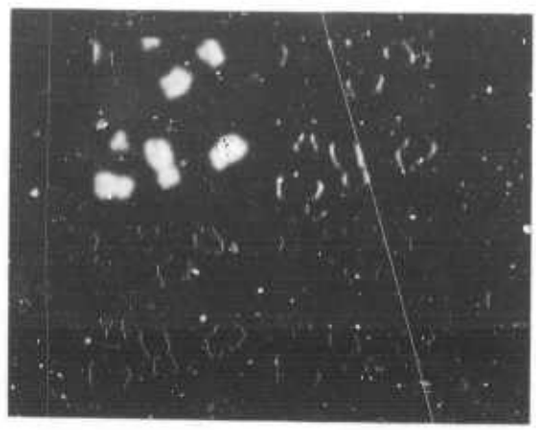
25. M. Shneier, Extracting linear features from images using pyramids, TR-855, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1980.

26. A. Y. Wu, T.-H. Hong, and A. Rosenfeld, Threshold selection using quadtrees, TR-886, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1980.

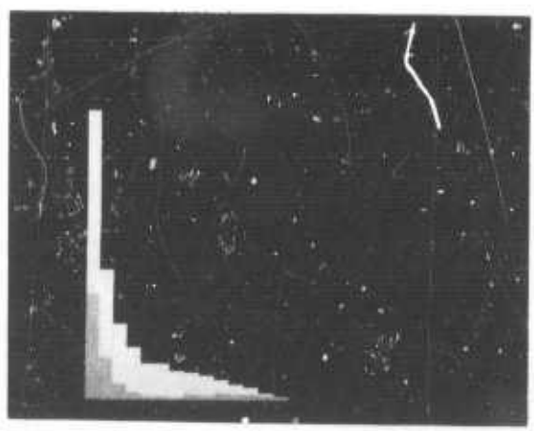
27. S. Ranade, A. Rosenfeld, and J. M. S. Prewitt, Use of quadtrees for image segmentation, TR-878, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, February 1980.

28. S. Ranade, Use of quadtrees for edge enhancement, TR-862, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, February 1980.

a

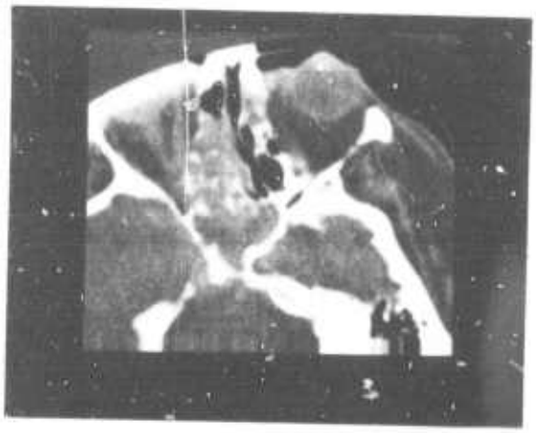


d



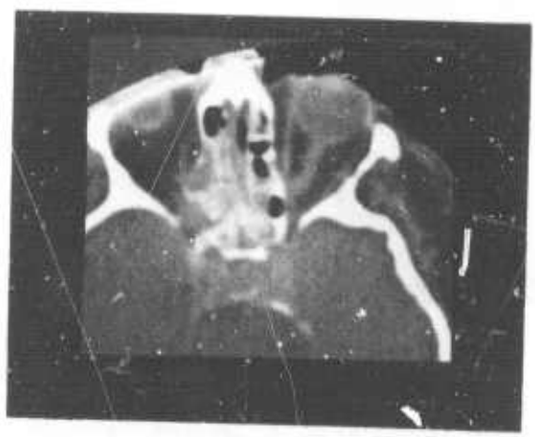
e

Figure 1. Nonmaximum suppression as an aid in edge detection. (a) Image; (b) digital (Sobel) gradient magnitudes; (c) results of suppressing nonmaxima in the gradient direction; (d) results of thresholding (b) at 6; (e) histograms of (b) and (c) superimposed.



a

b

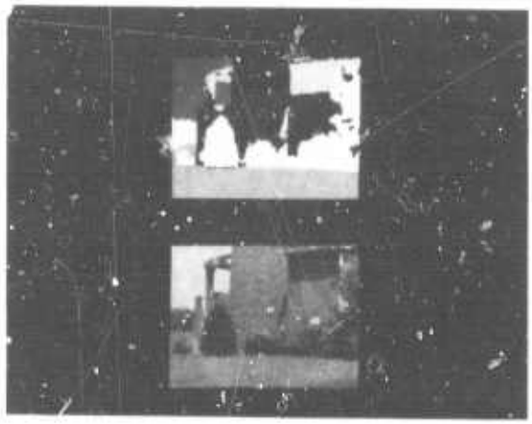


c

d

e

Figure 2. Surface detection in 3-d arrays. (a-c) Three consecutive cross-sections of a CT reconstruction; (d) results of applying the 2-d Prewitt operator to the middle cross-section; (e) results of applying a 3-d Prewitt operator to the three cross-sections.



c

Iteration of relaxation	Fraction of class correctly classified				
	1	2	3	4	5
0	.925	.938	.874	.057	.876
1	.933	.920	.871	.063	.872
2	.933	.921	.875	.067	.875
3	.933	.921	.876	.072	.874
4	.935	.921	.878	.157	.871
5	.937	.921	.880	.330	.841
6	.937	.921	.881	.413	.824
7	.938	.921	.881	.465	.820
8	.938	.921	.881	.500	.815

b

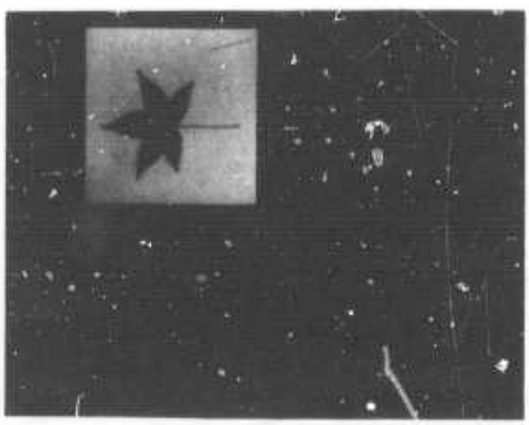
Class	Fraction classified as				
	1	2	3	4	5
1	.925	.003	.023	.003	.047
2	.011	.938	.028	.002	.021
3	.114	.002	.874	.000	.011
4	.019	.006	.003	.057	.914
5	.083	.005	.008	.025	.876

d

Iteration of medium filtering	Fraction of class correctly classified				
	1	2	3	4	5
0	.925	.938	.874	.057	.876
1	.897	.937	.915	.535	.866
2	.900	.940	.921	.536	.866
3	.901	.939	.921	.534	.867
4	.903	.937	.922	.531	.872

Figure 3. Segmentation based on gray level and local "busyness". (a) Image (bottom) and hand segmentation(top). (b) Confusion matrix for maximum-likelihood classification of the pixels into sky, brick, grass, bush and shadow, based on bivariate Gaussian fitting to the populations obtained by hand segmentation; note that the shadow class is mostly classified as bush. (c) Results of applying probablistic relaxation to initial classifications based on Gaussian fitting; note the gradual improvement. (d) Results of smoothing the busyness values by median filtering prior to clustering and classification; the improvement is immediate.

a



b

Figure 4. Corner detection in grayscale images. (a) Image; (b) results of "cornerity" computation.

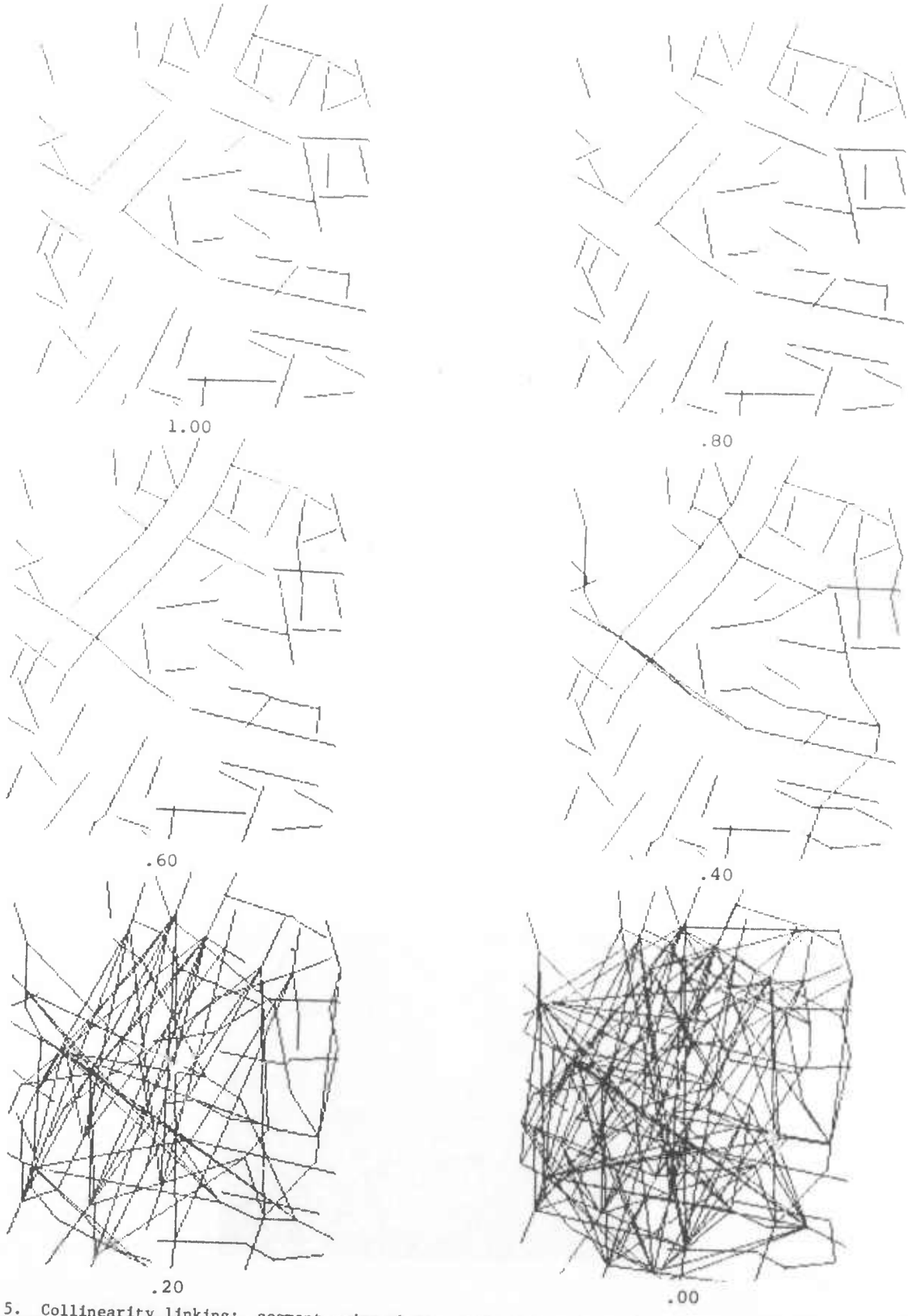


Figure 5. Collinearity linking: segment pairs whose collinearity merit is at least p, for p = 1, .8, .6, .4, .2, and 0.

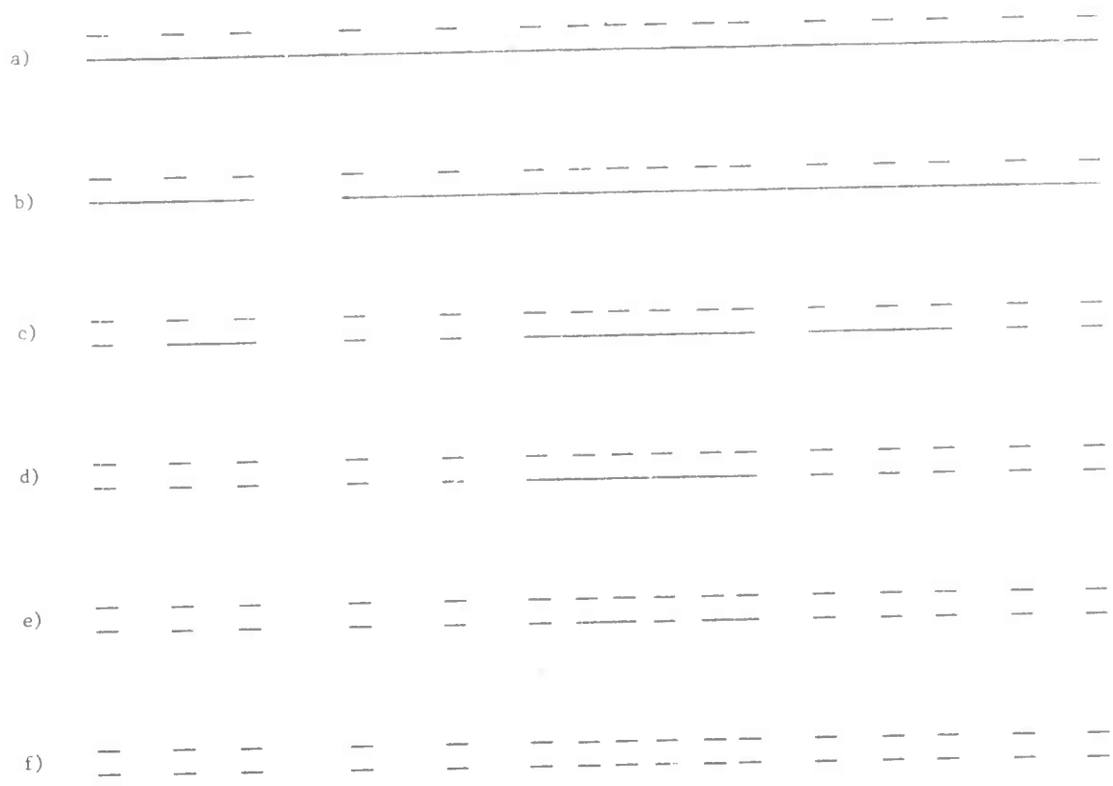
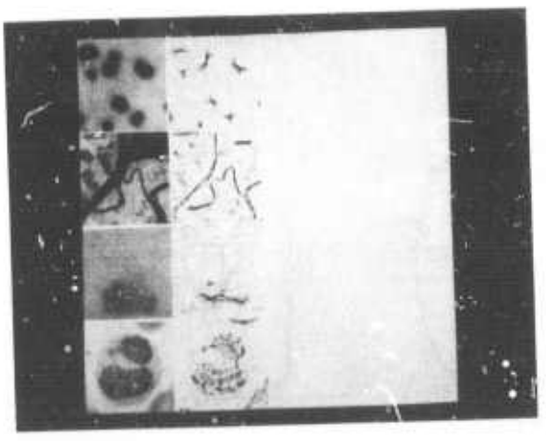
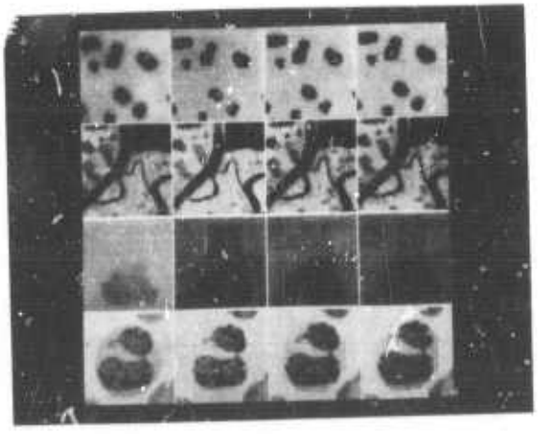


Figure 6. Collinear clusters of segments. The input is repeated on each line, with the results of clustering shown below it for successively higher values of a clustering parameter. For the value used in (d), the central cluster is linked together and no other segments are linked.



(a) (b)

Figure 7. The min-max medial axis transformation (MMMAT). (a) Images; (b) MMMATs.



(a) (b) (c) (d)

Figure 8. Reconstruction from the MMMAT. (a) Original images; (b-d) reconstructions from the one, two, and three largest increments at those points having values above the 25th percentile (189, 582, 226, and 462 out of 4096 pixels, in the four cases).

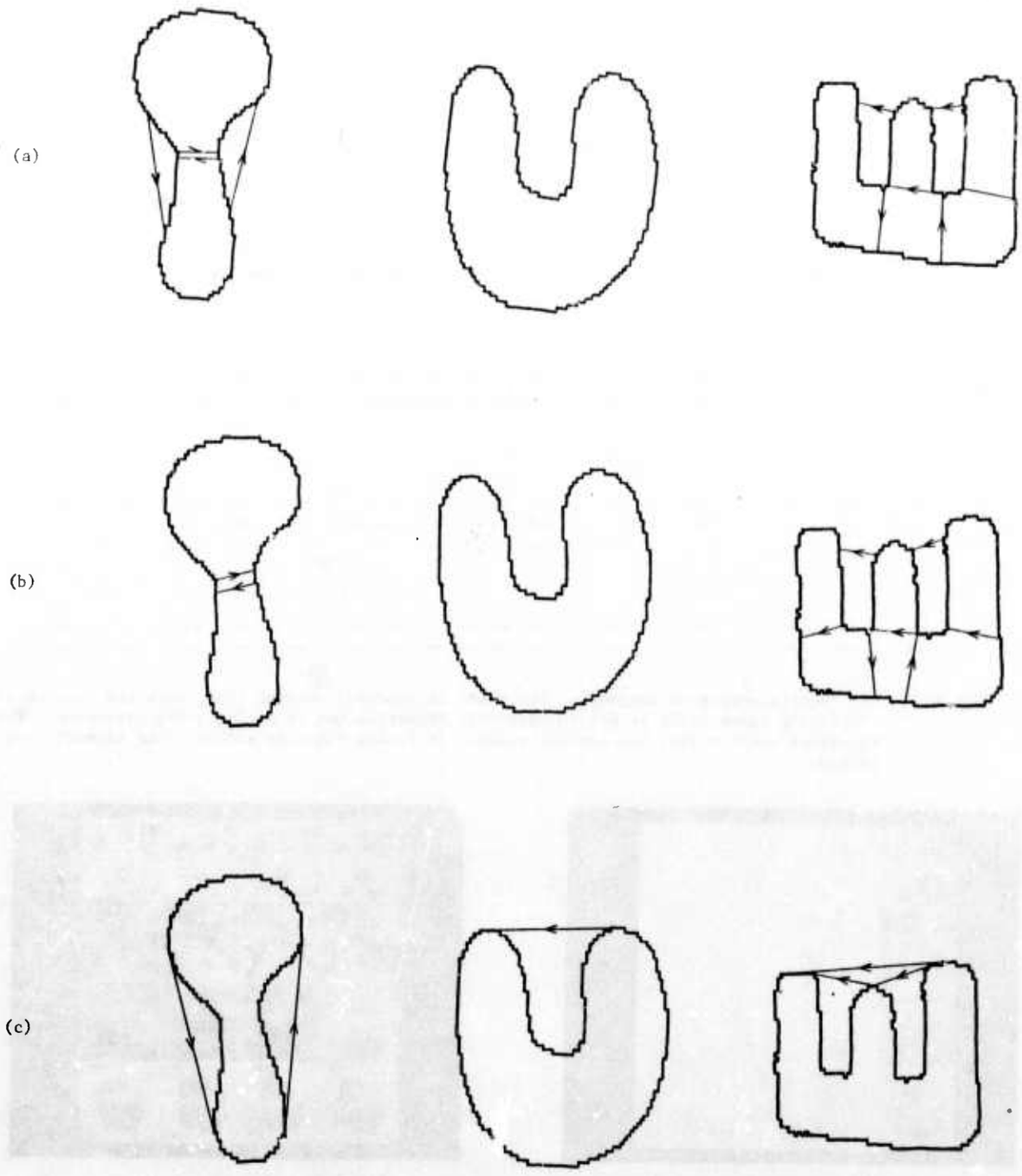


Figure 9. Shape segmentation based on extrema of arc/chord functions. (a) Maxima of area/chord^2; (b) maxima of arc/chord; (c) negative maxima of area.

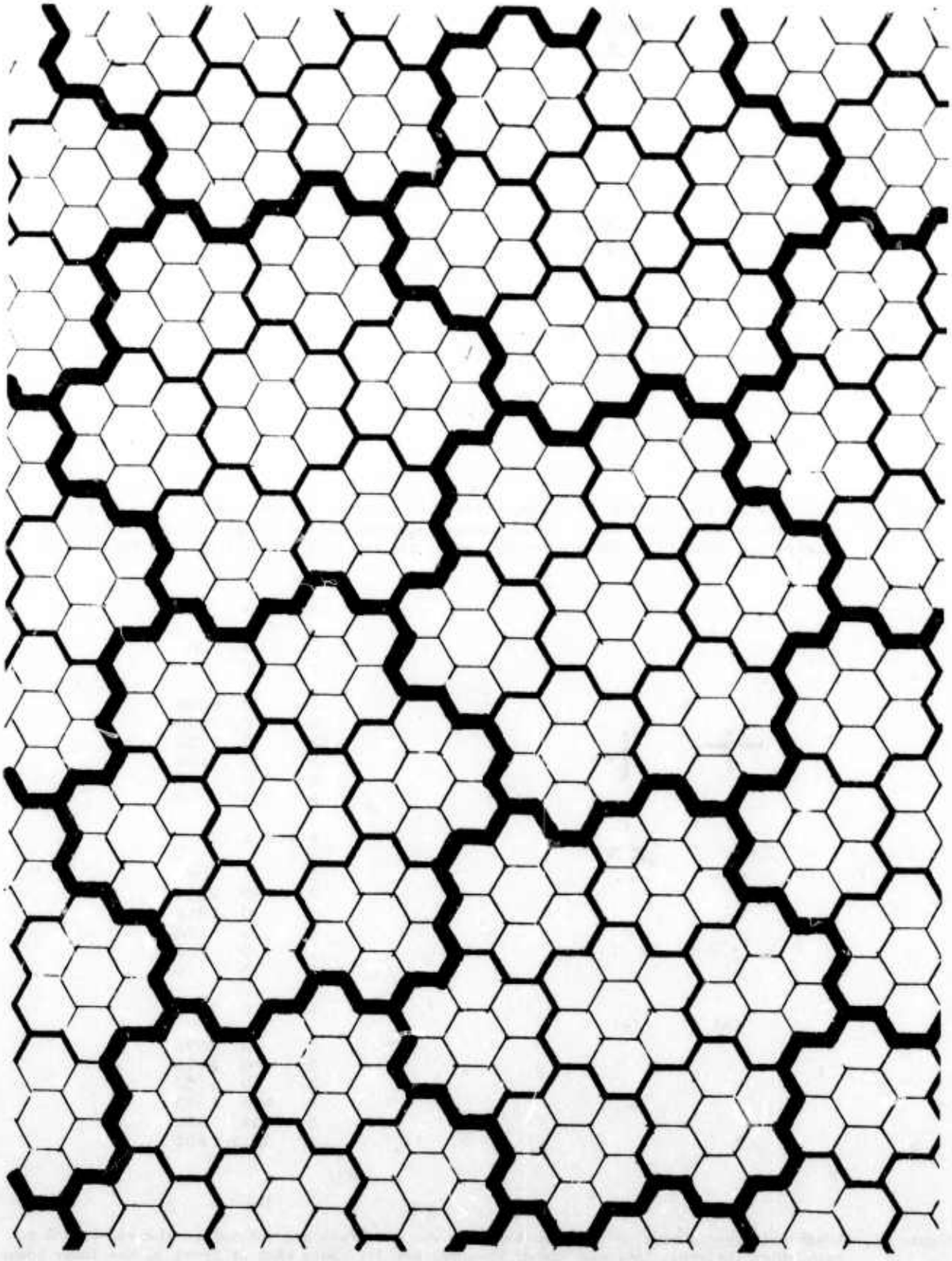


Figure 10. Hierarchical hexagonal grid (three levels).

(a)

(b): Level

2

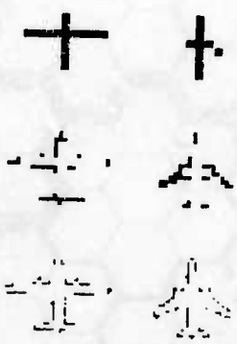
1

0

(c)

Level	No. of nodes	Area	Centroid coordinates
2	15	240	35.63,33.50
1	53	392	34.64,32.38
0	155	494	34.17,32.36

Figure 11. Approximating the centroid of a shape using its quadtree representation. (a) Airplane. (b) Black nodes at each level of the quadtree representation of (a), displayed as black blocks. (c) Cumulative number of nodes, area, and centroid coordinates as a function of level.



(a) (b)

a to a

L	LB	UB
1	0	4096
2	0	2048
3	0	1152
4	0	608
5	0	212
6	0	0

b to b

1	0	4096
2	0	2304
3	0	1216
4	0	592
5	0	200
6	0	0

a to b

1	0	4096
2	0	2816
3	0	1600
4	352	1392
5	504	900
6	601	601

(c)

Figure 12. Lower and upper bounds on the mismatch when two airplanes are matched to themselves and to each other (L=level, LB=lower bound, UB=upper bound). Note that at level 5, the lower bound on the mismatch to each other exceeds the upper bounds on the mismatches to themselves.

IMAGE UNDERSTANDING RESEARCH AT CMU: A Progress Report

John R. Kender and Raj Reddy
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

INTRODUCTION

Our efforts this past half-year have been to continue to develop an integrated demonstration system, and the image processing techniques that such a system implies. We can report progress on several fronts this past six months: advances in the theory and implementation of low- and middle-level image processing algorithms, basic system hardware enhancements, and software systems development. For much of this period we have been concentrating on creating and readapting algorithms for the VAX UNIX environment.

We continue to focus our efforts on three major tasks areas. The first concerns the understanding of two-dimensional aerial and satellite images of the Washington, D.C., area. We are examining cost-effective methods of matching the visual input against the symbolic map. Major effort has been applied to the difficult problems that occur when resolution is very good, including the modeling of buildings and the problems of shadows. Work on the three-dimensional analogue of this task--the understanding of color images of downtown Pittsburgh--has lead to further exploration of the relationships of surface texture to surface orientation. Development of high-speed convolution algorithms, applicable to both domains, has also started.

Our second area of concern is the development and evaluation of computer architectures for computer vision. Further work in this area awaits the delivery of the SPARC ultra-high speed signal processing computer (Allen, 1979), and the installation of the Programmable Sum of Products Operator chip (Eversole, et. al., 1980). Both are being developed on subcontract, to Control Data and Texas Instruments, respectively. We anticipate that a fair amount of software development will occur on their arrival.

It is in our third task area that we have devoted much of our energies: the development of user-friendly interactive aids for image processing applications. In addition to continuing work on our image data-base management programs (McKeown, 1979), we have written and modified many programs for the new hardware and software environment that a UNIX VAX with frame buffer provides. Some of our code should also make the IUS testbed facility a more efficient environment.

The following is a more detailed discussion of our recent work.

SYSTEMS

Software development has spanned many levels of software needs. We have worked on the facilities that are necessary for interprocess communication to be added to UNIX; this will aid the testbed. This has required substantial modifications and additions to the C language and its manual.

A bit higher, we have defined the necessary new image format, which more efficiently groups pixels into blocks instead of rows. We have coded the support and library routines that allow us to easily manipulate images. Although a given image can be hardware paged, we have provided programs to allow them to be software paged, for convenience and efficiency.

Our Grinnell frame buffer has been installed. We have begun to convert many of our system-level and user-level software from our multi-processor Image Understanding System to its new host, the UNIX VAX. The frame buffer will provide us with much-needed flexibility in the display of cultural data superimposed upon raw imagery; it also contains some local image processing capabilities. We are implementing a general-purpose user-level subroutine package for the frame buffer, including a system for generating the maps for its color mapping hardware.

User-level facilities under development include programs to display multiple images simultaneously, and programs for flexible menu display. Many low-level image operators and routines have been moved and adapted. Specific algorithms newly being brought up for the task domains are found in the next section.

At the level of existing full user systems, we have adapted the BROWSE information management system (Fox, et. al., 1979) to the VAX, as one of our interactive aids. The KIWI color-image segmentation system (Shafer, 1980), also highly interactive (as well as automatic) is in the process of being moved, too.

And, of course, we have reformated and copied much of our large library of images.

TASKS

Our efforts in the task domains have been both practical and theoretic.

Two separate efforts have explored algorithms to handle high-resolution aerial imagery. The first anticipates a system in which a symbolic map is accurate to the building level; the signal is matched to the symbols. Current design employs an image segmentation step, which will be followed by a chamfer-based matching step (Barrow, et. al., 1977). The segmenter is a type of region-grower. It appears that it may be easier to embed any heuristic knowledge necessary for this task into such a segmenter; further, it seems it would be easier to integrate such a segmenter into the matching step, if necessary.

Initial design of the segmenter was similar to that of Nagao, Matsuyama, and Ikeda (Nagao, et. al., 1978). However, it was necessary to augment and rewrite portions of the algorithm, apparently due to differences in resolution, and our lack of multispectral information. Several low-level operations were enhanced and tuned. Additionally, heuristics were added to guide the segmentation. Experimentation continues.

The second effort takes a somewhat different approach; it is line-based, and is intended to help build up the symbolic map. In this task, we have focused on the problem of extracting highly accurate line boundaries of buildings. As part of this system, we have experimented with various ways of obtaining and refining edge profiles. We believe that an understanding of profiles may lead to a new, efficient, and highly accurate method of locating and tracking extended line segments. We anticipate that the use of the lines themselves will also lead to efficient representations. This work intends to interactively build model descriptions of a city, directly from images contained in our database. Parallel efforts in using line descriptions for image-to-map matching have also begun.

Continuing research into the relations of textures to surface orientations has produced more theoretic results concerning the physical imaging process, the representation of surface orientation, and the problems of representing surfaces themselves (Kender, 1980). (Additionally, some aspects of the theory have been implemented and are being explored.)

A general method has been developed to find the camera parameters of focal length and focal point directly from features in the image. These parameters correspond roughly to the size of the lens and to the location of the center of the image before it was cropped. Until now, these parameters must have been known a priori before any image processing requiring absolute position could begin. Like all image understanding tasks, the method requires certain assumptions of the image; the method works best on scenes with sharp angles. The method suggests that topological relations between surfaces are far more important that exact ones; indirectly, it helps explain both the success and inextensibility of line-labeling schemes.

Additional work has revealed some insight into why the ground plane is so difficult to find in a single image under the conditions or assumptions of orthography. At least in one common instance, the problem of Necker reversal is compounded by a second ambiguity that leads to situations reminiscent of the drawings of Escher.

Since textures are often sparse, finding and representing the surfaces they delineate is a difficult task. It is often the case that a texture is largely transparent: consider a window screen, for example. Representing such textures is difficult, since it implies that the image itself is multivalued as far as distances and orientations at a given pixel are concerned. However, the problem is closely related to the problems of occlusions, and even of shadows (which often are common, but rarely are total).

A second area of theoretic endeavor (under ONR support) is exploring three-dimensional automatic concept formation. The goal is a program that is given a set of examples and non-examples of a concept representing a three-dimensional object (such as a chair). From these it generates a three-dimensional model of the concept. The approach describes objects as combinations of generalized cylinders. Carefully chosen properties of each object description are extracted, and compared with other objects to determine how best to describe the concept model. There are many issues that still need to be studied, among them devising heuristics for choosing promising features, and for merging features into good models.

Lastly, we are exploring techniques to decompose convolution masks into sequentially applied submasks that are more cost-effective. We hope matrix decompositions will allow us to quickly, but accurately, apply operators or match templates to our images; these expensive operations are required in all of our task domains.

REFERENCES

G. R. Allen, P. G. Juetten, "SPARC--Symbolic Processing Algorithm Research Computer," Proceedings of the ARPA Image Understanding Workshop, Science Applications Inc., Apr., 1979.

H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching," Proceedings of the International Joint Conference on Artificial Intelligence, 1977.

W. L. Eversole, and D. J. Mayer, "Programmable Sum of Products Operator," in this volume.

M. S. Fox, and A. J. Palay, "The BROWSE System: An Introduction," Proceedings of the Annual Conference of the American Society for Information Science, 1979.

J. R. Kender, "Shape from Texture," Ph.D. Thesis, Computer Science Dept., Carnegie-Mellon Univ., 1980 (in preparation).

D. M. McKeown, "Representations for Image Data Bases," Proceedings of the ARPA Image Understanding Workshop, Science Applications Inc., Nov., 1979.

M. Nagao, T. Matsuyama, and M. Ikeda, "Region Extraction and Shape Analysis of Aerial Photographs," Proceedings of the International Joint Conference on Pattern Recognition, 1978.

S. Shafer, and T. Kanade, "KIWI: A Flexible System for Region Segmentation," Technical Report, Carnegie-Mellon Univ., 1980 (in preparation).

UNDERSTANDING IMAGES AT MIT: REPRESENTATIVE PROGRESS

Kent Stevens, Keith Nishihara,
Brian Schunck, and the Staff

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

In this series of image understanding conference proceedings, we have stressed the issue of representation. In particular, we have described the development by Horn and his collaborators of the reflectance map, and the albedo image (in working with satellite images), and we have described the work of Marr and his group on the primal sketch, the 2 1/2-D sketch, and axis-based 3-D models as part of a comprehensive theory of recognition.

In the November, 1979 Proceedings, we reviewed our contributions to the design of adequate representations and enumerated techniques that we have devised to exploit them.

Here we review work by Horn's group on optical flow and work by Marr's group concerning zero-crossings, stereo, stereo hardware, and the contributions of texture gradients to the 2 1/2-D sketch.

Zero-Crossings and the Primal Sketch

Marr's group has devoted considerable attention to the theory of the *raw primal sketch* of the image, a primitive description of the intensity changes in terms of blobs, bars, edges, and terminations, which are characterized by position, orientation, contrast, and size. (The *full primal sketch* later emerges when local geometric relations are made explicit along with larger, more abstract descriptions of groupings, aggregations, and summarizing descriptions, e.g., of texture. Marr provided the foundations for the full primal sketch by specifying a number of grouping operations such as the so-called *theta aggregation*. Here we will report on recent progress on the earlier, raw primal sketch.)

Computing the raw primal sketch falls naturally into two parts: (i) the intensity changes at a set of different scales are first computed since intensity changes occur in natural images over a wide range of scales, and (ii) the descriptions that arise from these independent channels are then combined into the raw primal sketch of the image.

The use of multiple-scale zero-crossings in images filtered by convolution with the Laplacian of a two-dimensional Gaussian ($\nabla^2 G$) is now a central component of early vision computations [Marr & Poggio 1978; Marr, Poggio & Hildreth 1979]. This reflects two underlying requirements, the need to separate information in an image according to its scale, and to identify fixed locations on viewed surfaces. Peaks in the rate of intensity change correlate well with physical locations at the smallest scale and these peaks correspond to zeros in the Laplacian. Similar information at other scales can be obtained by first convolving the image with a Gaussian having a suitable space constant. Gaussian convolution has the property of removing high frequency information while preserving the local geometric structure of larger scale variations in intensity. The two steps, Gaussian convolution followed by the Laplacian, can be combined into a single operation of convolution with the Laplacian of the Gaussian, since $\nabla^2(G * I) = \nabla^2 G * I$. $\nabla^2 G$ filtered images are essentially the same as the difference of Gaussian filtered channels observed in the human visual system [Marr & Hildreth 1979].

The zero-crossings are represented by a set of oriented primitives called zero-crossing segments, each describing a piece of the contour whose intensity slope (rate at which the convolution changes across the segment) and local orientation is roughly uniform. Small, closed contours are represented as blobs, also with an associated orientation, average intensity slope, and size defined by their extent along a major and minor axis.

Zero-crossings and Sampling Theorems

Continuing work on Marr and Poggio's stereo theory has led to several computational results concerning (i) the relationship between mask size and resolving power at the level of zero-crossings, (ii) the sampling interval necessary to localize those zero-crossings reliably, and (iii) the sufficiency of the slopes at zero-crossings as a representation of $\nabla^2 G$ filtered images. The first two parts of this work started out as questions about the human visual system. Psychophysical experimentation has revealed the remarkable capabilities of the human visual system to resolve fine detail. Marr, Poggio, and Hildreth [1979] have shown that if the limit of resolution is determined by the smallest separation at which distinct zero-crossing contours can be obtained between two dots or lines, then the $\nabla^2 G$ mask of the smallest channel in the visual system must have a central excitatory diameter (w) of at most 1.5 minutes of arc. This is about an octave smaller than the smallest channel (w=4.38

minutes) measured psychophysically by Wilson and Bergen [1979]. This fifth channel is also consistent with the optical limitations of the eye which place a theoretical limit on resolution, and with known physiological data concerning midget ganglion cells in the retina which are believed to be driven by a single cone cell.

There is a second type of visual acuity, called hyperacuity, which refers to our ability to make accurate judgments requiring the localization of some visual feature to a resolution of a few seconds of arc, roughly one fifth the diameter of the smallest foveal cone cells [Westheimer 1976]. The input to the visual cortex has a sampling interval of about 1 minute of arc which is even coarser than the initial cone spacing. To explain this Marr, Poggio and Hildreth [1979] and Crick, Marr and Poggio [1980] have shown that zero-crossings can be localized to within a few seconds of arc by straightforward interpolation between values at the sample points.

These results have relevance to both the study of human vision and the development of practical machine vision systems. To make the properties of ∇^2G filtered images more precise, Keith Nishihara has been studying the constraints placed on ∇^2G filtered images by boundary conditions at (i) their zero-crossings and (ii) regularly spaced sample points. The objective of the first is to determine the degree to which the slopes at the zero-crossings determine the overall filtered signal. It is important to understand how different two filtered signals can be and still satisfy the same zero-crossing boundary conditions. The nature of ∇^2G filtering does not allow a uniqueness theorem such as Logan's [Logan 1977]. Nevertheless, it is possible to show that the difference between two filtered images which have the same zero-crossings and the same slopes at those zero-crossings is bounded in a useful way which depends on (i) the smallest distance between zero-crossings at the point in question and (ii) the range of magnitudes allowed in the original images. These results are consistent with, and strengthen, the earlier empirical results by Nishihara which showed that a good approximation to a ∇^2G filtered image can be reconstructed from just the slopes at its zero-crossings [Winston 1979]. The techniques used to obtain these results can also be used to bound the difference between two ∇^2G filtered images having the same values at regularly spaced intervals.

Stereo Implementation and Stereo Hardware

The implementation by Eric Grimson of Marr and Poggio's stereo algorithm [Marr & Poggio 1978] has been tested successfully on a wider range of natural images and this work has led to further refinements of the implementation [Grimson & Marr 1979, Grimson 1980, see also Grimson's paper in these Proceedings]. Much of his effort is presently directed toward the problem of interpolation between the contours of known depth provided by the stereo program. The basic components of the computer implementation are straightforward and can be implemented efficiently in hardware.

Noble Larson has completed the design of hardware to compute ∇^2G convolutions at near video rates using the Hughes CCD convolver chip [Nudd et. al. 1979]. The hardware construction is underway and should be completed soon after the chip becomes available from Hughes. Larson and Nishihara are also working on the hardware design for the stereo matcher which will work off of the outputs of two convolvers, one for each of the stereo images, and it should also operate at near video rates. The stereo hardware has two components, matching and statistics checking. The matching step involves determining the number of possible matches (zero-crossings with the same sign) in a neighborhood in the left image that is twice as wide as the positive part of ∇^2G. The neighborhood is centered on the position corresponding to a zero-crossing in the right image (see Grimson's paper for further details). If there is only one possible match it is accepted as a candidate match and its position relative to the position of the zero-crossing in the other image is passed on to the statistics module. The statistics module receives this information along with bits indicating whether there was a zero-crossing to be matched and whether or not at least one possible match was found as a raster scan from the matching module. The Marr and Poggio algorithm allows candidate matches to be accepted only if the ratio of matches found to number of zero-crossings requiring matches in a neighborhood about the candidate match is greater than what would be expected if unrelated images were being compared. This computation will be accomplished by buffering the last 20 or so lines of data from the matcher and computing a running ratio for a 20 by 20 neighborhood at the current raster position.

Texture Gradients

The information content of "texture gradients" has been re-examined by Stevens [1980]. Texture gradients are systematic variations in the density, size, and other measures of projected surface texture. It is generally expected that these texture variations encode information about the shape of the surface either in the form of surface orientation, or distance, or perhaps both. Various mathematical relations have been proposed between quantities in the image texture such as density and 3-D quantities such as distance or slant. However most of these relations are not useful since they embody assumptions (in the form of geometric restrictions, e.g., for global planarity) which are seldom satisfied in natural scenes. The problems of computing surface orientation and distance were examined in turn. Each computation is assumed to have local support in the image. A principle result is that while both computations share several common limiting factors, the distance computation is often more robust and accurate than the surface orientation computation.

The perspective projection may be usefully thought of as comprising two independent transformations to any patch of surface texture: scaling and foreshortening. Scaling is due to distance, foreshortening is due to surface orientation. A decomposition of the problems of computing distance and surface orientation is therefore

suggested: when computing distance, the texture measure (the specific numeric quantity extracted from each locality of the image texture) should vary only with scaling; when computing surface orientation, the measure should vary only with foreshortening.

One consequence of this is that texture density is not a useful measure for computing distance or surface orientation, since it varies with both scaling and foreshortening. This explains the observation made by some psychologists that a pure density gradient (e.g., of dots) is ineffective in suggesting a definite 3-D surface. If density is not a useful measure for computing either distance or surface orientation in general, what texture measures should we choose?

First consider the distance computation. Distant features on a surface project to a smaller size than those that are closer, provided the features are physically the same size. Therefore a smooth surface of uniform texture presents a continuously varying scale from which distance up to a multiplicative constant might be recovered. What remains to be made precise is the notion of "size" or "scale" in terms of real images. The appropriate measure for the distance computation, keeping in mind the measure should vary only with distance and not foreshortening, are termed *characteristic dimensions* and correspond to nonforeshortened dimensions on the surface. Distance up to a scale factor may be computed from the reciprocals of the characteristic dimensions, assuming that the corresponding physical dimensions on the surface are uniform. Since we assume that no *a priori* knowledge of the physical makeup of the surface is available at the point in visual processing at which the depth map is computed, the computational problem centers on choosing the characteristic dimensions, for a natural image presents a wealth of potential useful dimensions in any locality of the image. Fortunately, characteristic dimensions may be defined in the image by the following geometric properties: they are locally parallel, oriented perpendicular to the texture gradient, and are parallel to the orientation of greatest texture regularity. Analysis of the local image texture can then identify the characteristic dimensions, and their reciprocals specify the depth map.

The precision and accuracy of the depth map is limited by the uniformity across the surface of the physical dimensions that correspond to characteristic dimensions. Evidence of uniformity is present in the textured image; this evidence would be useful in restricting the distance computation to those instances where the depth map would be likely correct. The visual evidence for uniformity of the actual surface texture is both local and global. Locally the texture must project as regular (e.g., the characteristic dimensions must have small variance locally) and globally the texture must be qualitatively similar. Examples of similarity measures might be color and intensity statistics, coarse shape description and other measures that are roughly invariant over perspective projection. The local regularity and global qualitative similarity together allow one to deduce global uniformity, for constraints on the physical texture that are so strong as to restrict the surface markings to a small range of sizes in any locality are often independent of the position of the markings on the surface. (For example, oak leaves strewn across a yard are qualitatively similar and have similar sizes. The global uniformity in leaf size is a consequence of how leaves grow and is independent of how they are distributed across the ground.)

Surface orientation is also believed to be computable from the texture gradient. There are actually two paths that might lead to a representation of local surface orientation [Marr 1977] where the primitives specify the slant and tilt [Stevens 1980] of each visible patch of surface. The first path is to first compute a depth map, e.g. by the method just described, then to compute the slant and tilt from the gradient of distance. This indirect path is successful only when there is significant scale variation in the image and fails for surfaces in orthographic projection. (If the surface is relatively distant -- the variation in distance to the surface is insignificant relative to the mean distance -- then the projection is effectively orthographic, or parallel projection. Surface cannot be computed from the depth map in those cases because the depth map would falsely indicate a flat surface in the frontal plane.) The other path is to attempt to compute surface orientation directly from the image. Accordingly, the texture measures used in doing so should vary with foreshortening but not vary with scaling. However such measures are difficult to interpret unless the particular foreshortening function is known which relates the measure to surface slant. Furthermore, successive occlusion associated with viewing texture which lies in relief relative to the mean surface level acts to confound the apparent foreshortening. Slant is therefore difficult to compute. However the tilt may be computed as the orientation of the characteristic dimensions.

Atmospheric Modeling

Turning now to Horn's work, recall that in the previous proceedings, we listed the following uses for synthetic images: automated generation of shaded relief maps, generation of low-level, obliquely-viewed images, generation of special maps that bring out particular terrain features, classification of ground cover for crop prediction, matching images to terrain data for satellite navigation, and making maps for automatic or semiautomatic change detection.

In general, four factors must be considered when making synthetic images for these purposes. They are: (i) imaging geometry - the projection of the viewed scene onto the image, (ii) incident illumination - the intensities and distribution of light sources, (iii) surface photometry - the way a surface reflects light, and (iv) surface topography - the shape of things in the scene.

Synthetic images that are to mimic real ones obtained from spacecraft require attention to a fifth factor: the atmosphere attenuates visual signals, scatters spurious light into the viewing port of the satellite, and illuminates the ground as a large, diffuse light source. Sjoberg's paper in these Proceedings describes work on modeling such effects.

Optical Flow

We are beginning research to develop an algorithm that will determine the motion of objects in a scene from a sequence of views of the scene taken closely spaced in time. In particular, we will determine the flow of motion of the image intensities that comprise the successive views of the scene, restricting our attention to the situation where the viewer is stationary and objects in the scene are possibly in motion. More precisely speaking, we are concerned with the situation where a scene with moving components is projected onto a stationary image plane and the changing images are quantized in both space and time. The problem is that given samples of the projection of the scene onto the image plane, we would like to construct an estimate of the velocity flow of objects across the image plane and we would like the density of the reconstruction, that is, the number of points at which the flow is determined, to be at least as great as the pixel density in the successive image frames. The resulting velocity field, derived from the successive images by using the information in the image intensities to maximum advantage, provides a complete picture of the motion of objects in the scene.

It must be stressed that we wish to develop an algorithm that uses the information in the image intensities to the fullest advantage; we are unwilling to incorporate higher level knowledge about the constituents of the scene into our algorithm. We will only use the constraints that can be derived from the physics of image irradiance and the relationships between the image irradiances from successive views of a scene. The starting point of this work will be an equation of constraint derived by noting that the change in image intensity for small displacements of objects in the scene will be zero. Specifically, let $E(x,y,t)$ denote the image intensity at point (x,y) in the image frame taken at time t. Then the requirement that the total change in image intensity be zero translates to

$$\frac{dE}{dt} = 0,$$

which by expanding according to the chain rule becomes

$$\frac{dE}{dt} = \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0.$$

Denoting the x and y components of the velocity field by u and v, respectively, we get the following linear equation in the unknown variables u and v

$$\frac{\partial E}{\partial x}u + \frac{\partial E}{\partial y}v + \frac{\partial E}{\partial t} = 0,$$

which is valid at each point in the successive image frames under the assumption that the displacement of moving objects in successive frames is sufficiently small that changes in image irradiance due to variation in orientation of the surface elements relative to the viewer and sources of illumination may be neglected. Given that the effect of such variations is negligible, the validity of the equation does not depend on the reflectance properties of the objects in the scene as long as the objects are opaque.

The partial derivatives in the linear constant-coefficient equation may be computed at each point in each successive image frame independent of the particular values of u and v. So at each point in an image frame we will have a constraint on the possible direction and magnitude of the motion. This constraint may be viewed as a line in (u, v) space along which the actual motion may lie. The line of constraint in velocity space is always perpendicular to the gradient of the image intensity at the corresponding point in the image. As the gradient of the image intensity varies across an object that is undergoing pure translational motion in a direction that is roughly perpendicular to the direction of view, the orientations of the corresponding velocity constraint lines will vary, but all lines will intersect at the point in velocity space that is the actual velocity of the object. Since we only need two linearly independent constraint lines to uniquely determine the motion, there is, in principle, more than enough information in the image intensities to determine the velocity field.

In practice, the coefficients of the equation will be difficult to compute accurately by numerical methods. Any scheme for combining the constraints in a straightforward manner will suffer from insufficient accuracy. Even if the coefficients could be accurately computed, it is not sufficient to merely intersect the constraints derived from objects that are undergoing rotational motion or translational motion along a line that is primarily parallel to the direction of view, since adjacent points in the image would not have equal velocities.

Nevertheless, the problem of computing the velocity flow requires computing a consistent solution to a large set of linear equations that overdetermine the solution. The computational problems are not insurmountable. We intend to continue investigating schemes for combining the motion contraints inherent at each point in successive image frames into a complete and accurate estimate of the motion of objects in the scene.

Bibliography

For an extended discussion of MIT work on Image Understanding, see Volume 2 of *Artificial Intelligence: an MIT Perspective*, edited by Patrick H. Winston and Richard H. Brown, MIT Press, Cambridge, Massachusetts, 1979.

F. H. C. Crick, D. Marr, and T. Poggio, "An Information Processing Approach to Understanding the Visual Cortex." To appear in the N.R.P. Symposium *The Cerebral Cortex*, Schmidt, F. O. & Worden, F.G., eds., 1980.

W. E. L. Grimson, "A Computational Theory of Human Stereo Vision", Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980 (in preparation).

W. E. L. Grimson and D. Marr, "A Computer Implementation of a Theory of Human Stereo Vision," *Proceedings of ARPA Image Understanding Workshop*, 41-45, April 1979.

Berthold K. P. Horn, "The Position of the Sun," Working Paper 162, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

Berthold K. P. Horn and Brett L. Bachman, "Using Synthetic Images to Register Real Images with Surface Models," AIM-437, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977.

Berthold K. P. Horn and Robert W. Sjoberg, "Calculating the Reflectance Map," AIM-495, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978. Also in *Applied Optics*, June, 1979.

Berthold K. P. Horn and Robert J. Woodham, "LANDSAT MSS Coordinate Transformations," AIM-465, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

Berthold K. P. Horn and Robert J. Woodham, "Destriping Satellite Images," AIM-467, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 1978.

Berthold K. P. Horn, Robert J. Woodham, and William M. Silver, "Determining Shape and Reflectance using Multiple Images," AIM-490, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

B.F. Logan, "Information in the Zero-crossings of Bandpass Signals," *Bell System Technical Journal 56*, 487-510, 1977.

D. Marr, "Representing Visual Information," AIM-415, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977.

D. Marr and E. Hildreth, "Theory of Edge Detection," AIM-518, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1979.

D. Marr and Poggio, T. "A Computational Theory of Human Stereo Vision," AIM-451, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

D. Marr, T. Poggio and E. Hildreth, "Evidence for a Fifth, Smaller Channel in Early Human Vision," AIM-541, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1979.

D. Marr, T. Poggio and S. Ullman, "Bandpass Channels, Zero-crossings, and Early Visual Information Processing," AIM-491, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

G. R. Nudd, S. D. Fouse, T. A. Nussmeier, and P. A. Nygaard, "Development of Custom-Designed Integrated Circuits for Image Understanding," *Proceedings: Image Understanding Workshop*, 1-9. November 1979.

Kent A. Stevens, "Surface Perception from Local Analysis of Texture and Contour," TR-512, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980.

G. Westheimer, "Diffraction Theory and Visual Hyperacuity," *Am. J. Optometry and Physiol. Optics 53*, 362-364, 1976.

H. R. Wilson and J. R. Bergen, "A Four Mechanism Model for Threshold Spatial Vision," *Vision Res. 19*, 19-32, 1979.

P. H. Winston, "MIT Progress in Understanding Images," *Proceedings: Image Understanding Workshop*, 25-35, April 1979.

PROGRESS AT THE RCCHESTER IMAGE UNDERSTANDING PROJECT

J. A. Feldman
K. R. Sloan, Jr.

The University of Rochester
Rochester, New York 14627

1. Model Refinement

1.1. Constraint Networks and Procedural Description

One important goal of the Rochester Vision Project is to investigate a generalized representation of complex objects by semantic networks. In our formulation these include procedural invocation in which an executive procedure chooses worker procedures to perform a job not just on the basis of input/output behavior (as traditional pattern- directed invocation does), but also taking into account cost/benefit estimates and perhaps other information as well. This scheme is motivated by the desire to have the advantages of declarative knowledge about what is doable (the descriptions) along with the advantages of procedural knowledge about how to do it (the workers). The declarative, descriptive component will allow conviences such as the modular addition of procedural knowledge. The main research issue is to decide what exactly needs to be known about worker procedures, and how to express that in a useful and uniform manner. This must also be coordinated with the use of relational constraints [Russell and Brown, 1978]. A recent paper at Rochester exploring aspects of these issues is [Lantz et al., 1978].

1.2. Decision Theory

The use of decision theory not only as an abstract model of intelligent perception but as a practical tool to maximize computational benefit/cost is being investigated in the context of procedural image [illegible]. This work continues in the tradition of Bolles, Sproull, and Garvey, and ultimately we hope to extend some of their results to deal with formal problems that more closely approximate the sorts of vision problems encountered in our particular applications. Ballard (see Section 9) uses decision theory techniques to choose the most economical method (assuring adequate accuracy) of locating anatomical structures in large-format images.

2. Applications in Biomedicine

Work has been underway at Rochester for several years on developing techniques for reliably detecting specific visual features, even in the presence of considerable noise. Our work has been based on generalizations of the Hough technique, which accumulates evidence for straight lines at various slope and intercept values using an accumulator array. For some time, we have been successfully employing extended Hough techniques to locate second-order curves like elliptical sections and circles. In the last six months we have been able to extend these techniques to handle a broad class of features [Ballard, 1978a]. There is reason to believe that these noise-resistant feature identification methods can be combined with our constraint graph techniques (cf. Section 1) to yield a robust and general analyzer for industrial site images.

3. Application in Aerial Image Analysis

The three-level organization of image analysis (strategist, executive, worker) and a further exploration of useful procedural description mechanisms were first applied to photointerpretation work in [Lantz et al., 1978]. The object is to use the sorts of knowledge- based inferencing used by skilled photointerpreters, along with models inspired by photointerpretation keys for identifying small industries, to do reliable and flexible identification of a few types of small industrial installations.

A second phase of experimentation was based on the analysis of selected industrial sites using locally acquired aerial imagery. We have now acquired and digitized a sample image from the Defense Mapping Agency and are working on the structure of our third generation system. The current plan is to rely heavily on the

general techniques described in Sections 1 and 2 above.

4. Image Encoding and Transmission

4.1 Hierarchical Image Encodings

Communication of images, and information about images is an important part of any image understanding project. We have been investigating the use of various hierarchical image encodings. One of the image transmission schemes we have investigated is closely related to "pyramid" data structures. We have demonstrated that high resolution raster images can be effectively transmitted over relatively low-bandwidth lines by sending a series of low resolution approximations, which converge to the final image [Sloan and Tanimoto, 1978].

A second hierarchical encoding is described elsewhere [Ballard, 1979a]. A Strip Tree is an elegant encoding for curves which represents both open curves (linear features) and closed curves (areas) in a uniform manner. Strip Trees are closed under intersection and union and operations on them can be carried out at different resolutions. These properties make them a nearly ideal representation for map data bases.

4.2 Composition and Re-interpretation of Images

It is often convenient to specify an image in terms of the combination of several existing images, rather than transmit an entire new image. The combination or re-interpretation may sometimes be performed with relatively simple hardware devices. We have developed and implemented several such techniques based on the "video lookup table" supplied with our Grinnell GMR-26 display [Sloan and Brown, 1979]. These techniques are currently being used to overlay map features on aerial images, display three-dimensional surfaces under quickly varying lighting conditions, and show short, repetitive motion sequences.

5. Component Building

5.1. Hardware

The Grinnell GMR-26 display device is DMA-interfaced to an Eclipse computer, and has been invaluable as an output device for our experiments. An Optronics Colorscan C-4100 drum scanner is on site and interfaces to the Vision Eclipse.

Both Eclipse computers are fully configured and have been running effectively with our distributed system software. A VAX 11/780 (purchased with non-DoD funds) is operating and has been integrated into the local network. A new, larger capacity Eclipse has been added to the gateway configuration, giving greater capacity and reliability. We are expecting several additinal personal compters and a laser printer later this year.

5.2. Software

Advanced system software support is now used routinely, and more is under development. Communications protocols and distributed computing packages [Feldman 1978, Sheininger and Sabbah 1977, Selfridge 1979, Sloan 1978] have been developed to allow access to the GMR-26 through the local ALTO computers or the remote PDP-10, to achieve reliable transmission between distributed processes, to produce graphics and halftone images on ALTO screens from the PDP-10, and to allow file transfer and telnet to the Arpanet. At Rochester, the RIG messae is the lingua franca that allows processes on remote machines to command the GMR-26, perform file manipulations, and other operations. Some of our work has been utilized by other image understanding groups, most extensively at SRI. We have been working closely with other IU contractors (particularly CMU) to develop a uniform communication facility for use in the testbed.

A comprehensive library of vision routines [Sloan 1977-78] has been developed, centralized, documented, and incorporated into the NEXUS system. They allow interactive users a wide range of image-processing and display (graphics, halftone, color and B&W TV) capabilities. A program to acquire images from the Optronics scanner and package them according to our Raster Image File Format [Selfridge and Sloan, 1979] has been developed and is in routine use.

6. Motion Understanding

Understanding motion pictures has always presented an unusually difficult problem to computer vision efforts. The compelling gestalt induced in humans by moving objects is not well understood, and so there is little leverage on the immediate problems resulting from the large mass of data in multi- frame images. We began on a pared-down version of the problem which nevertheless offers an interesting set of perceptual phenomena to model. The domain is multi- frame images of animal motion; initial research is being carried out on sequential images of points of light attached to joints. A

detailed progress report was presented at the last IU Workshop.

7. Parallel Algorithm Development

This is a new development in our laboratory which resulted from a combination of several separate prior efforts. The main idea is that there is a systematic duality between our generalized Hough techniques (cf. 2.) and the highly parallel models of computation we are developing for describing animal vision [Feldman, 1979]. This has given rise to some preliminary ideas for highly parallel implementations of image understanding algorithms which seem promising. These ideas are being pursued in cooperation with the VLSI and theory of computation groups at Rochester.

8. Texture

Textural areas can be thought of as those parts of an image where segmentation based on normal similarity measures fails. Meaningful analysis of textured areas must include discrimination between different textures and detection of parts of the same texture. The similarity of textures which are identical except for a scale change, a rotation, or a different range of intensities must be recognized.

We approach the texture problem by dividing texture regions into meaningful sub-elements of similar intensity sample points, then using rotation- and scale-invariant shape measures to characterize these regions and finally determining spatial relationships among our sub-elements. By using a decision tree program structure, easily discriminated textures are separated quickly, and more complex textural structure is extracted only when necessary [Maleson, Brown, and Feldman, 1977]. A major report on this work will be available this summer.

9. Applications in Biomedicine

The model-directed finding of ribs in chest radiographs [Ballard, 1978] provides an illustration of the use of the Rochester Vision System, incorporating procedure description, utility measures, and top-down, model-directed perception. The object here is to cope with large amounts of possibly low-quality data without undue processing time by depending on a declarative model of anatomical structures, described procedural knowledge about how to locate them, and an executive which uses decision theory to control the image-understanding process. A prototype complete analysis sytem is now being developed.

A novel and uniform method of describing arbitrary functions on the unit sphere (which define "museum-viewable" volumes) is under investigation, with immediate application to anatomical structures [Schudy and Ballard, 1979]. The idea is related to the well-known Fourier descriptions of two-dimensional shape. Volumes are modelled and described as the leading coefficients in certain spherical harmonic expansions of the volume functions. This method also allows least squared error fitting of volumes in coefficient space, which interfaces nicely with routines that locate the three-dimensional boundaries of volumes in image data.

Applications of generalized cylinders [Agin, 1972] previously have been limited to simple cross sections. We use B-splines as an embedding for generalized cylinders [Shani, 1979]. This allows an efficient realization of the original notion of generalized cylinders as arbitrary cross sections about a space curve.

REFERENCES

Agin, A.P., Representation and description of curved objects, Stanford AI Project Memo AIM-173 (and Ph.D. thesis), October, 1972.

Ballard, D.H., Model-directed detection of ribs in chest radiographs, TR11, Computer Science Department, University of Rochester, March 1978.

Ballard, D.H., Generalizing the Hough Transform to Detect Arbitrary Shapes, TR55, Computer Science Department, University of Rochester, October, 1979 (a); submitted to Pattern Recognition.

Ballard, D.H., Strip Trees: A Hierarchical Representation for Map Features, Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, August, 1979 (b).

Barrow, H.G., et al., Interactive aids for cartography and photo interpretation, Semiannual Technical Report, Artificial Intelligence Center, SRI International, November 1977.

Feldman, J.A., A distributed information processing model of visual memory, TR52, Computer Science Department, University of Rochester, December, 1979.

Feldman, J.A., Systems support for advanced image understanding. DARPA Semiannual Technical Report, May 1978.

Lantz, K.A., Brown, C.M. and Ballard, D.H., Model-driven vision using procedure description: Motivation and application to photointerpretation and medical diagnosis, 22nd International Symposium of the Society of Photo-optica Instrumentation Engineers, San Diego, Ca., August, 1978.

Maleson, J.T., Brown, C.M. and Feldman, J.A., Understanding natural texture, DARPA Image Understanding Workshop, October, 1977.

Russell, D.M., Where do I look now?: Modeling and inferring object locations by constraints, Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, August, 1979.

Russell, D.M. and C.M. Brown, Representing and using locational constraints in aerial imagery, Image Understanding Workshop, November, 1978.

Scheininger, U., and Sabbah, D., The display process, Internal Memo, Computer Science Department, University of Rochester, December 1977.

Schudy, R.G. and D.H. Ballard, Towards an anatomical model of heart motion as seen in 4-D cardiac ultrasound data, IEEE Conf. on Computer-Aided Analysis of Radiological Images, June, 1979.

Selfridge, P.G., A flexible data structure for accessory image information, TR45, Computer Science Department, University of Rochester, May, 1979.

Selfridge, P. and Sloan, Jr., K.R., "Raster Image File Format (RIFF): An Aproach to Problems in Image Management," TR 52, Computer Science Department, University of Rochester, May, 1979.

Sloan, Jr., K.R., Rochester vision library documentation, Internal Memos, Computer Science Department, University of Rochester, 1977 - 1978.

Sloan, Jr., K.R., and Brown, C.M., "Color Map Techniques," Computer Graphics and Image Processing, 10, 4, August, 1979.

Sloan, Jr., K.R., and Tanimoto, S.L., "Progressive Refinement of Raster Images," TR?, Computer Science Department, University of Rochester, November, 197?.

Shani, U., B-splines as an embedding for generalized cylinders, Computer Science Department, University of Rochester, forthcoming.

SPATIAL UNDERSTANDING

Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

We have introduced a representation mechanism in ACRONYM for specific and generic objects and partially specified scenes. The mechanism relies on specialization by constraints on a class of variables called quantifiers. The specialization mechanism is used in implementing constraints in interpretation associated with alternate candidate model matches. The rule language of ACRONYM has been entirely revised to simplify rule sets.

A powerful system has been developed for determining parameters of object models in interpretations of image descriptions. This enables subsequent information gathering and detailed testing of small object structures.

INTRODUCTION

We describe extensions to ACRONYM, the model-based interpretation system. ACRONYM includes a geometric modeling subsystem, a geometric reasoning subsystem, and subsystems for *description, prediction, and interpretation.* The geometric modeling system supports a high level language for object models as structures of generalized cylinders. A rule-based geometric reasoning subsystem is used by the other subsystems. We use the word description to mean a structure instantiated from representation elements, a structure of relations and primitives at all levels. By description process we mean building up structural descriptions from image level to the level of volume or surface. By prediction, we mean synthesizing structural descriptions at the level of observables (edges, ribbons, surfaces in stereo) or closely related level. Prediction and description are closely complementary and interact closely in that they share the same knowledge base. We report work in all these parts of ACRONYM, but particularly in geometric reasoning and interpretation with partially constrained object models and scenes, and in determining model parameters in interpretation.

ACRONYM

ACRONYM interprets in the domain of volumes; we are working on two approaches to go from images to volumes. We have developed the predictive approach furthest, to make a first tentative identification between image features and observables of objects, then to make detailed verification of those interpretations. Lowe has built an important element of the detailed verification stage, a program which solves for model parameters given an identification of image descriptions with observables of a model ([6]). Given accurate model parameters, gathering of additional information and detailed search for small features can be carried out. The system provides a natural way of using partial knowledge as constraints on model parameters, e.g. the fact that an aircraft is on the ground implies its support points are in a horizontal plane. The system determines model parameters and a coordinate transform which relates the object to the observer frame. It includes articulated models and constraint relations on parameters. It sets up a search system automatically with appropriate parameters. The solution is expressed in terms of image features which are lines and points; lines are a natural output from image descriptions. Newton-Raphson search converges in less than 6 iterations typically, with each iteration executing in about 20 msec on a DEC KL10 in compiled MACLISP. We are extending the system to evaluate which parameters can be determined, to set up constraint conditions on those parameters which can be determined, to set up initial estimates for search parameters, and to evaluate which measurements should be made to determine parameters which have not been sufficiently constrained.

GEOMETRIC REASONING

Previously, the rule language was quite limited. The data base for rules consisted of assertion triples. Because most geometric information was in Object and Observability graphs, most manipulations took place outside the rule mechanism in the form of side effects which were not transparent to the rule based reasoning. Brooks has made a completely new, much generalized rule language which works with Object and Observability graphs directly. Rule sets are much simpler and clearer, much less is hidden, and the system has much more power. The rule base for aircraft recognition has been redone in the new rule language.

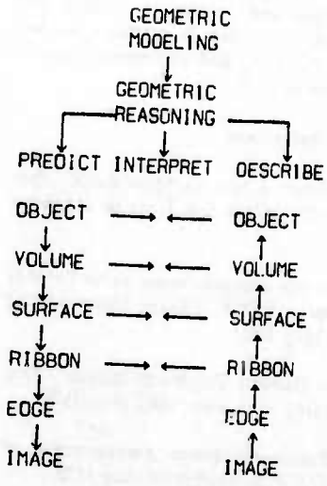
Brooks has introduced a new mechanism for dealing with partially specified objects and scenes ([2]). ACRONYM incorporates *restriction nodes* which represent the following with one mechanism. 1. instances or subclasses of models; 2. multiple aspects of a single object; and 3. multiple candidate ribbons matching a single object and multiple object interpretations for single image ribbons. They all are

represented as specializations of models. Partially specified information is represented by *quantifiers*, identifiers whose values satisfy systems of constraints. They provide a natural representation for symmetry.

The interpreter phase uses quantifiers in matching image descriptions with object models. When the interpreter makes tentative matches of image features such as ribbons, it constructs constraints on quantifiers of candidate instances of object models. At the same time, it checks consistency of constraint systems. This provides a powerful mechanism for making globally consistent interpretations of local constraints. The interpreter is being changed to a rule-based control structure, in order to satisfy these new requirements. Figure 1 shows one aspect of ACRONYM structure.

The Predictor and Planner has new capabilities for prediction of quasi-invariant observables using symbolic manipulation of coordinate transforms of partially-specified objects and scenes. Thus, the system uses not just a pre-packaged set of quasi-invariants, but any quasi-invariants which it can deduce from special information about the particular case, including dynamic information which arises in the course of matching.

FIGURE 1
ACRONYM



A Geometric Editor, called MODITOR (model editor) has been added to ACRONYM. MODITOR operates at the textual level, traversing symbolic models. A system for generating stereo pairs for crossed viewing has been added. Work is underway toward a hidden surface display algorithm to be used for human visualization and for the Predictor and Planner. A priority graph scheme enables a large part of display computation to be done once at the time of modeling, instead of each time a display is generated.

We are in the process of converting our vision system to MACLISP from SAIL, to integrate all capabilities in ACRONYM. The conversion is intended also to facilitate portability to the VAX system, as well as moving toward compatibility with a segment of the Image Understanding community. We advocate ARPA development of a MACLISP+LISPM portable system to complement ADA. We are planning to adopt graphics and image handling protocols compatible with some subset of IU groups. SRI's proposals [Quam] look like good choices. We have transported Nevatia and Babu's edge finding and curve linking system to the WAITS time-sharing system [Nevatia].

STEREO VISION

Arnold and Binford are working out detailed geometric constraints for stereo correspondence. Any correspondence between two images represents a particular evaluation function and search procedure among all the possible correspondences. CDC minimized mismatch along epipolar lines on a line to line basis, using dynamic programming with a heuristically chosen evaluation function We are designing an evaluation function from first principles, while considering solution procedures which include interline constraints. The dynamic programming procedure requires significant improvements: First, it does not have solutions with overhang because two sequences of edges must have monotonic correspondence. Second, it should include interline constraints; we have generalized the procedure to include interline constraints, however it requires prohibitive computation cost.

It is well known that two views of n indistinguishable points have n-squared ambiguous correspondences. Along a single epipolar line, after all local distinctions between edges are made, the n-squared ambiguity remains among all indistinguishable subsets. If two views are interpreted by mapping onto a single underlying surface, then stretching and cutting in the image correspond to tilting, bending, and folding of the surface. With n indistinguishable edges there appear to be of order n*N surface interpretations (where N is the number of occlusions at depth discontinuities), given reasonable interpretation assumptions. That is a significant reduction, small enough to be enumerable. However, there are powerful constraints which reduce that number. An important general constraint is occlusion. Arnold has derived an image condition necessary for occlusion; we intend to find a related sufficient condition. We have found additional strong local constraints on surface interpretations. These will be described in a forthcoming paper.

One of the strongest constraints is continuity of edges from line to line. Arnold used edge continuity previously [Arnold 77] with considerable success without these additional geometric constraints. Liebes is developing constraints for cultural scenes, for horizontal and vertical surfaces, particularly planes and cylindrical surfaces. Vertical surfaces give special problems in establishing correspondence, however there are strong special case constraints for these surfaces. Those constraints are useful also in establishing correspondence in cultural areas for passive navigation using linear features.

Baker describes an edge-based stereo mapping system which searches for consistent edge correspondences line-by-line

along epipolar lines ([1]). At then rejects pairings which violate interline continuity. Implementation of the system was motivated by a desire for speed; it executes in about 30 seconds for 256x256 images. Edges are obtained by zero crossings of 1x7 and 7x1 bar masks. They are paired on the basis of matching contrast or intensity on one side of the edge. For each edge there are a set of possible matches. Formerly, a branch and bound search process was used to establish correspondence. That has been replaced by a Viterbi algorithm dynamic programming method. The Viterbi algorithm is much faster. It returns only a single best solution however, which may be error sensitive. That is, the locally optimal solution for single lines may not be globally consistent. A later stage removes some of these errors. The program uses a coarse to fine search procedure like the binary search correlator introduced by ([8]), however limited to single epipolar lines. Branch and bound search and dynamic programming both sought solutions which maximize the number of edges which matched; among solutions with equal numbers of edge matches, they sought solutions which minimize the sum of squared errors of intensities of intervals to either side. Pairings for individual epipolar lines are tested for consistency by testing depth continuity. The consistency procedure follows connected edges in both images, calculating local mean and standard deviation of disparity, and removing pairings by a sequence of tests on disparity.

Clarkson and Binford are evaluating feasibility of improving the solution program for the stereo camera transform ([3]) to make it more stable and robust, i.e. less sensitive to errors of mismatched points, to improve speed of solution, and to make a single solution program for both degenerate and non-degenerate cases. The solution must determine a rotation of one camera coordinate system relative to the other (3 parameters) and a translation unit vector of one camera center relative to the other (2 parameters). Rays through the two lens centers to a single space point are coplanar. This provides one constraint per point. Translation parameters are poorly determined in the degenerate case with little relief compared to camera distance, We are investigating solutions which separate out subspace solutions. Clarkson has found a condition using pairs of correspondences to determine the rotation independent of the translation. He is evaluating whether that constraint system has a computationally effective solution. We are also considering using corresponding lines, which are natural for edge operators, and considering alternative parameterizations which may lead to simplified solutions.

EDGE-BASED DESCRIPTION

We are developing an improved edge finding and curve linking program based on extensions of the approach of the Binford-Horn system ([5]). At that time we considered the following problems and solutions for them: 1. Smooth shading gives continuous areas of false edges with gradient operators. Solution: "lateral inhibition", difference from the local average. 2. Low contrast edges are difficult to find. Solution: directional derivatives of laterally inhibited signal; sequential detection edge-linking algorithm. 3. Gradient operators do not provide accurate edge estimates (a function is flat near its maximum). Solution: interpolate edges from zero-crossings of the laterally inhibited signal. 4. Texture and surface marks provide spurious edges. Solution: Use directional derivatives and use a non-linear measure (gaussian residuals) in curve linking. We are working toward improved localization in position and angle, curve linking which is isotropic in angle, and improved treatment of spurious marks.

Marr and coworkers have given a valuable alternative motivation for the use of zero crossings, based on analogy with Logan's theorem on characterizing one-dimensional waveforms from zero crossings ([7]). They provide powerful insight into the role of zero crossings in natural stereo vision and edge finding.

PLANS

Research will continue to extend the interpretation capabilities of ACRONYM. The module for determination of model parameters will be integrated into ACRONYM and it will be extended to deal with partial information. The representation mechanism for quantifiers and specialization by constraints will be completed. Both will be used in experiments in interpretation with aircraft and vehicles.

Work is beginning on extending descriptive parts of ACRONYM beyond edges and regions to surfaces and volumes, using generalized cylinders. In addition, we will integrate stereo with ACRONYM and implement a part of the large descriptive system of ([9]).

References

[1] Baker, H.H., "Edge-Based Stereo Correspondence", *Proc. ARPA Image Understanding Workshop*, Univ of Md, May, 1980.

[2] Brooks, R., "Representing and Reasoning about Partially Specified Scenes", *Proc. ARPA Image Understanding Workshop*, Univ of Md, May 1980.

[3] Gennery, D., "Stereo Camera Transform Solution" *Proc. ARPA Image Understanding Workshop*, USC, Nov. 1979.

[4] Henderson, R., "Automatic Stereo Reconstruction of Man-Made Targets"; *SPIE Proc.* Huntsville, Aug 1979.

[5] Horn, B.K P., "The Binford-Horn Edge Finder", *MIT AI Memo 285*, revised December 1973.

[6] Lowe, D., "Solving for the Parameters of Object Models from Image Descriptions"; *Proc. ARPA Image Understanding Workshop*, Univ of Md, May. 1980.

[7] D.Marr, E.Hildreth; "Theory of Edge Detection"; AI Memo 518, AI Lab MIT, April 1979. D.Marr, T. Poggio, "A Theory of Human Stereo Vision," *AI Memo 451*, MIT, Nov 1977. W.E.L. Grimson and D. Marr, "A Computer Implementation of a Theory of Human Stereo Vision"; *Proc ARPA Image Understanding Workshop*, Palo Alto, April 1979.

[8] Moravec, H. P., "Towards Automatic Visual Obstacle Avoidance", *Proc IJCAI-5*, MIT, Boston, Aug 1977.

[9] Nevatia, R., "Structured Descriptions of Complex Curved Objects for Recognition and Visual Memory", A I Lab Stanford University, *Memo AIM-250*, CS-464, ADA003486, October 1974.

[10] Nevatia, R.,and Babu, K.R., "Linear Feature Extraction and Description", *Proc. of IJCAI-79*, Tokyo, Aug. 1979, 639-641.

THE SRI IMAGE UNDERSTANDING PROGRAM

M. A. Fischler (Principal Investigator)
SRI International
Menlo Park, California 94025

INTRODUCTION

Research at SRI International under the ARPA Image Understanding Program was initiated to investigate ways in which diverse sources of knowledge might be brought to bear on the problem of analyzing and interpreting aerial images. The initial phase of research was exploratory and identified various means for exploiting knowledge in processing aerial photographs for such military applications as cartography, intelligence, weapon guidance, and targeting. A key concept is the use of a generalized digital map to guide the process of image analysis. The results of this earlier work were integrated into an interactive computer system called "Hawkeye" [1]. This system provides necessary basic facilities for a wide range of tasks in cartography and photo interpretation, and it provides a framework within which other applications can be readily demonstrated.

Research subsequently focused on development of a program capable of expert performance in a specific task domain--road monitoring. The primary objective of this ongoing research is to build a computer system that "understands" the nature of roads and road events. It is intended that it be capable of performing such tasks as:

(1) Finding roads in aerial imagery.

(2) Distinguishing vehicles on roads from shadows, signposts, road markings, etc.

(3) Comparing multiple images and symholic information pertaining to the same road segment, and deciding whether significant changes have occurred.

The general approach, and details of technical progress on developing the components of the Road Expert are contained in References [2-6]. We are now integrating these separate components into a coherent system that facilitates testing and evaluation and will be in a form suitable for transfer to the ARPA/DMA Integrated Demonstration System ("testbed"). Plans for the Road Expert demonstration system are presented in Reference [7].

Recently, we have initiated major efforts in two new directions. The first is in support of a joint ARPA/DMA program to provide a framework for demonstrating the applicability of image understanding research (from throughout the entire IU community) to military problems in general, and to the problems of automated cartography in particular. Our plans and progress in this effort are described later in this paper.

The second effort is to broaden the scope and generality of our image understanding research--specifically in the areas of 3-D terrain understanding, perceptual reasoning, and image description and matching. A complementary research program (described in Reference [7]), jointly supported by ARPA and NSF, augments these investigations by focusing on fundamental computational principles underlying early stages of visual processing in both man and machines.

THE ARPA/DMA INTEGRATED DEMONSTRATION SYSTEM ("TESTBED")

Overview

ARPA and DMA have jointly established an integrated demonstration system ("testhed"), with SRI as the integrating contractor. The system, which is being developed at SRI, is intended to be used for demonstrating and evaluating the applicability of IU research to cartography. For this purpose it will have a user interface that simulates the environment of a cartographic work station consisting of a computer with CRT terminal, an image display with track ball, and a digitizing tablet. With the exception of image digitization, which for most purposes will be performed off-line by DMA, the system will support all major steps in map making with a continuously evolving degree of automation.

Initially the system will allow interactive creation and editing of digital maps in a fashion similar to Hawkeye [1]. Existing maps, for example, can be overlaid on new imagery, edited, and extended, using a variety of interactive aids for tracing linear features and modeling objects. The system will also allow remote execution (over the ARPANET) of automated and semi-automated techniques developed by IU contractors. Those techniques whose utility and reliability justify tighter integration will then be incorporated into the system.

The system will also be usable as a research facility, providing the IU and DMA communities with access to the most advanced tools available. This, plus the existence of compatibility standards for programs and data, will encourage building upon the work of others, allowing more ambitious projects to be undertaken. Furthermore, the availability of common data sets will promote more systematic evaluation of competing techniques.

The above objectives share the requirement for a very flexible architecture so that contributions developed in a diverse community can be fully

utilized. Integration support must be provided for a wide range of contributions, from primitive image processing techniques (e.g., an edge follower) to stand-alone subsystems (e.g., an expert system for terrain modeling). Language and operating system support must be sufficiently broad to encompass programs running under the multitude of systems used throughout the community. These systems can be expected to change continuously during the testbed's lifetime.

To meet these requirements, the system will be configured as a library of application modules that accepts input and deposits results in a shared global data base. For normal research and development, modules can be directly controlled in an interactive environment via the keyboard and graphical devices. For demonstrations a front-end process is interposed, simulating the environment of a cartographic work station similar to Hawkeye. This interface facilitates communication with the system via menus (e.g., ZOG) or limited natural language (e.g., LIFER, RITA) and includes a "help" facility.

Each application module performs a well-defined, high- or low-level task. Modules are independently compiled so each can be implemented in different languages and can reside on different processors. Modules interact with each other by means of a standard interfacing mechanism, resembling a procedure call. Details of how control is actually passed will, of course, vary, depending upon the level of module integration (same address space, same processor, or remote processor).

Most data interactions will be affected by accessing the shared data base. Modules operate on common data from the data base and deposit their results back into the data base, where they will be available for display or subsequent processing by other modules.

The data base is accessed via a uniform query language, which enforces compatibility and maintains integrity without constraining a module's internal representation. Modules need not know the source of the data they use nor who will use their results. For example, a program that needs the locations of edges in image X will look in the data base; if they are not there, the program requests the use of an edge-locator module which will deposit results tagged as "edges for image X" in the data base, where they will remain available for future use. The data base is thus the key to modularity in a large integrated system.

The VAX 11/780 has been selected as the main testbed machine. All tightly integrated parts of the system will be resident there. All other IU machines will be viewed uniformly as remote ARPANET hosts, including SRI's KL-10. However, the SRI KL will have a high-speed channel to the VAX via a shared disk, so application modules running there will incur minimal overhead.

SRI is building a core system on the VAX. It will include a data base and some general system utilities, such as display servers, data base manager, work station front end, and an ARPANET gateway.

Development of application modules will proceed in parallel at all IU sites. Each site will maintain local copies of relevant parts of the official data base (e.g., imagery) needed to develop and demonstrate their routines. Each site will also be able to call remotely resident modules over the network, using the gateway mechanism.

SRI will use the ARPANET to exercise application modules in the context of the core system. As utility and performance justify, modules will be imported to run at SRI on our KL or VAX. As the final demonstration takes form, critical modules may be recoded to integrate efficiently with the core VAX environment.

Having completed the system definition, our time schedule is to do the detailed design and construction of the core system in 1980, integrate application modules provided by the IU community in 1981, and evaluate and possibly extend the system in 1982.

Progress

The VAX 11/780 computer system has been installed at SRI. The U.C. Berkeley UNIX and DEC/VMS operating systems have been obtained and subjected to benchmark testing to determine their relative merits with respect to our special needs. Since UNIX has been agreed upon as the ARPA community standard, our use of VMS will be compatible with, and fully support, a UNIX operating environment. Processing environment standards to facilitate the community-wide effort are required for the success of the testbed effort and have received a great deal of attention on our part in recent months. A document describing our detailed plans and proposals in this area is currently available. (Requesters can obtain this file over the ARPANET from the file <TESTBEDDOC>INTEGRATION.DOC@SRI-KL.)

RESEARCH ACCOMPLISHMENTS

Two of our recent research accomplishments are described in papers published in these proceedings.

A paper by Fischler and Bolles [8] introduces a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and thus is ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. New results are derived for the minimum number of landmarks needed to obtain a

solution, and algorithms are given for computing these minimum-landmark solutions in closed form. These results form the basis for an automatic system that can solve the LDP under severe viewing and analysis conditions. Implementation details and computational examples are also presented.

A paper by Quam [9] addresses problems associated with the access of elements of large multidimensional arrays when the order of access is either unpredictable or is orthogonal to the conventional order of array storage. Large arrays (specifically arrays which are larger than the physical memory available to store them) must be accessed either by the virtual memory system of the computer and operating system or by direct input and output of blocks of the array to a file system. In either case, the direct result of an inappropriate order of reference to the elements of the array is the very time-consuming movement of data between levels in the memory hierarchy, often costing factors of three orders of magnitude in algorithm performance.

The access to elements of large arrays is decomposed into three steps: transformation of the subscript values of an n-dimensional array into the element number in a one-dimensional virtual array; mapping of virtual array position to physical memory position, and access to the array element in physical memory. The virtual-to-physical mapping step is unnecessary on computer systems with sufficiently large virtual addresa spaces.

A subscript transformation is proposed that is believed to solve most of the order-of-access problems associated with conventional array storage. This transformation is a based on an additive decomposition of the calculation of element number in the array into the sum of a set of integer functions applied to the set of subscripts as follows:

$$\text{element-number}(i,j,k,\ldots) = fi(i) + fj(j) + fk(k) + \ldots$$

The choices for the transformation functiona which minimize access time to the elements of the arrsy depend on the chsracteristics of the memory hierarchy of the computer system and the order of accesses to the elements of the array. It is conjectured thst there are easily obtained models for system and algorithm access characteristics, from which a pragmatically optimum choice can be made for the subscript trsnsformation functions.

The use of tsbles to evaluste the functions fi snd fj makea the implementstion very efficient usiag conventional computers. When the array accesses are made in an order inappropriate to conventional array atorage order, this scheme requires far less time than for conventional arrsy sccessesing schemes otherwise the accessing times are compsrable. The semantics of s set procedures for array scceas, array crestion, snd the association of srrays with file names is defined. For computer syatems with insufficient virtual memory, such as the PDP-10, a software virtual to physical mapping scheme is used. Implementations to access pixels of large images stored as two-dimensional arrays of n bits per element for the VAX and PDP-10 series computers are presented.

ACKNOWLEDGEMENT

Contributors to the SRI Image Understanding Program include: G. J. Agin, S. Barnard, H. G. Barrow, R. C. Bolles, M. A. Fischler, T. D. Garvey, G. A. Jirak, D. L. Kashtan, L. H. Quam, J. M. Tenenbaum, A. P. Witkin, and H. C. Wolf.

REFERENCES

1. H. G. Barrow et al., "Interactive Aids for Cartography and Photo Interpretation. Progress Report, October 1977," in Proceedings: Image Understanding Workshop, pp. 111-127 (October 1977).

2. M. A. Fischler et al., "Interactive Aids for Cartography and Photo Interpretation," Semiannual Technical Report, SRI Project 5300, SRI International, Menlo Park, California (October 1978 and May 1979).

3. L. Qusm, "Road Tracking and Anomaly Detection," in Proceedings: Image Understanding Workshop, pp. 51-55 (May 1978).

4. R. C. Bolles et al., "The SRI Road Expert: Image-to-Database Correspondence," in Proceedings: Image Understanding Workshop, pp. 163-174 (November 1978).

5. G. J. Agin, "Knowledge-Based Detection and Classification of Vehicles and Other Objects in Aerial Road Images," in Proceedings: Image Understanding Workshop, pp. 66-71 (April 1979).

6. M. A. Fischler, J. M. Tenenbaum, snd H. C. Wolf, "Detection of Roads snd Linear Structures in Aerial Imagery," in Proceedings: Imsge Understanding Workshop (November 1979).

7. M. A. Fischler, "The SRI Image Understanding Program," in Proceedings: Image Understanding Workshop (November 1979).

8. M. A. Fischler snd R. C. Bolles, "Random Sample Gonsensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Gartography," in Proceedings: Image Understanding Workshop (Aprll 1980).

9. L. Qusm, "A Storage Representation for Efficient Access to Large, Multi-Dimensional Arrays," in Proceedings: Image Understanding Workshop (April 1980).

PROGRESS IN IMAGE UNDERSTANDING RESEARCH AT USC

Ramakant Nevatia
and
Alexander A. Sawchuk

Image Processing Institute
Electrical Engineering and Computer Science Departments
University of Southern California
Los Angeles, California 90007

We have continued work at various levels of our IU system and started to evaluate techniques for applications to DMA supplied images. These activities are described in more detail in our semiannual technical report [1], and the following contains only a brief abstraction.

IMAGE MATCHING

Matching of an image to a symbolic map, or a symbolic description of another image, is central for the tasks of map updating and change detection. In the past we have described results using aerial images of areas such as San Francisco, Stockton, San Diego, etc. [2]. These previous techniques used a simple matching scheme, with each element in one description being matched to the best corresponding element in the other description, and not allowing for any revision based on the matching of other neighboring elements. We have now incorporated a relaxation matching algorithm. This algorithm is different from those used by Rosenfeld and associates [3], in use of a well defined optimization criterion. Details of this algorithm are described in a separate paper in these proceedings [4].

TEXTURE ANALYSIS

We have continued development of our structural texture analysis techniques. The analysis uses micro-edges detected in a texture and derives repetition pattern characteristics of these edges. Our previous presentations have described techniques for determining the width of texture primitives and a repetition period, if any. Our new techniques are also able to extract the length of the primitives and thus describe the shapes of the primitives. The usefulness of these descriptions for recognition of natural textures is currently being tested.

TEXTURE SYNTHESIS

We have several ongoing projects in synthesis of natural textures from a stochastic model using few parameters. The techniques include auto-regressive modeling with conditional expectations and algebraic reconstruction techniques. Models for texture synthesis are also useful for texture analysis, as they validate the sufficiency of the models used for analysis.

SEGMENTATION

We are trying to use the texture analysis techniques to aid in scene segmentation. Texture features can be used as intensity features for segmentation. However, difficulties arise because texture features must be measured over a window assumed to contain a single texture only. Also, texture features have many components and should be treated as a vector. Faugeras and Lee give some preliminary results in [1].

Another segmentation project is to develop techniques for segmenting images that have unimodal intensity (or color) histograms. This happens typically when the image consists mostly of a large background region, with small but significant other regions. Faugeras and Bhanu have developed a gradient relaxation technique to modify the histogram to bring out the peaks corresponding to the small regions [1]. Currently, this technique is applicable if only two types of regions are present in the image.

HARDWARE IMPLEMENTATION

In continuing work with Hughes Research Laboratories, Malibu, California, we are investigating the use of VLSI technology for hardware implementation of IU algorithms. We have chosen to investigate the following algorithms initially:

i) Nevatia-Babu Line Finder [5]
ii) Ohlander Region Segmentor [6]
iii) Laws Texture Analysis System [7]

The choice of the above three algorithms was based on their computation intensive nature, their use for a broad range of problems and experience with a large number of images for the first two. Also these algorithms are largely local and hence easier to implement in VLSI hardware, where reducing interconnections is important. Further, the three algorithms have common kernels, such as convolution, but also require different subsequent processing. A

study of there should provide valuable feedback on the feasibility of hardware implementation for a large class of algorithms.

At this time, no decision on algorithms for actual implementation has been made and opinions of the IU community are invited on the suitability of the proposed algorithms as well as suggestions for other algorithms.

REFERENCES

1. R. Nevatia and A.A. Sawchuk, "Semiannual Technical Report," USCIPI Report #960, March 1980.

2. R. Nevatia and K. Price, "Locating Structures in Aerial Images," Proceedings of ARPA Image Understanding Workshop, Palo Alto, Ca., October 1977.

3. A. Rosenfeld, R.A. Hummel and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. on Systems, Man & Cybernetics, SMC-6, No. 6, pp. 420-453, June 1976.

4. O. Faugeras and K. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," in these proceedings.

5. R. Nevatia and K.R. Babu, "Linear Feature Extraction," Proceedings of the ARPA Image Understanding Workshop, Pittsburgh, Pa., Nov. 1978, pp. 73-78.

6. R. Ohlander, K. Price and R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Computer Graphics and Image Processing, Vol. 8, 1978, pp. 313-333.

7. K. Laws, "Textured Image Segmentation," USCIPI Report #940, January 1980.

SESSION II

TECHNICAL PAPERS

TOWARD THE RECOGNITION OF CULTURAL FEATURES

Mohamad Tavakoli

Computer Vision Laboratory
Computer Science Center, University of Maryland
College Park, MD 20742*

ABSTRACT

The goal of this research is to find a method for recognition of cultural features such as roads and buildings extracted from aerial photographs. The approach involves several successive stages of grouping of linear features. In this process, it is highly desirable to avoid firm decisions at any stage, but rather to make fuzzy or "probabilistic" decisions whenever possible, thus deferring commitments until they are confirmed by other evidence. The decisions at each stage are based on as much information as possible, and each stage uses an appropriate type of data representation. In this report, the stages of the feature extraction process, as they are presently conceived, are described, and examples of results obtained at the first few stages are given.

INTRODUCTION

Cultural features often contrast with their surrounds, and are usually bounded by sharp, locally straight edges. Thus in order to find cultural features, edge detection and line finding techniques can be used as a starting point. Such techniques have been studied for many years; partial surveys may be found in [1-5].

Several considerations have led us to use an edge-based approach at the pixel level. We first use local operators to estimate the magnitude and direction of the gradient at each point. We then use an iterative process at the pixel level to adjust the magnitudes and directions. See [6] for more details. The approach to feature extraction at the University of Southern California [4] is also edge-based, but it involves a one-step process of non-maximum suppression and thresholding, rather than an iterative, quantitative process. The USC approach is thus computationally cheaper, but it is probably more likely to make errors.

After extracting edges at the pixel level, we want to construct a more global data representation. This will allow us to use more knowledge about cultural features. In order to obtain a more global representation a global straightness criterion is used in defining connected components of edge pixels by requiring each pixel's direction to be close to the average direction of the already accepted pixels [6]. This breaks up smooth curves into segments having relatively low net change in slope from one end to the other.

We now have a set of edge segments, with each of which we can associate various properties. At this stage local properties are used to define a figure of merit vector representing initial guesses for object interpretation. Using this vector, one can define the initial probability assignment.

After the above stage, we use knowledge about the linear features to find compatible and anti-parallel pieces and group the edge segments. Finally, we update the probabilities of each line segment based on these groups of line segments and the physical descriptions of the objects.

In what follows, we present a detailed description of the design of some of the low level grouping operators.

INITIAL PROBABILITY ASSIGNMENTS

As already mentioned, a global straightness criterion is used to construct the edge segments. We associate various properties with each segment, including its length, average strength, etc., as well as properties of the gray levels on the two sides of the segment's constituent edge pixels. At this level there are many edge segments which do not belong to objects such as roads or buildings. They are edges of other types of objects or simply noise. For simplicity these edges are called "other edges."

One of the most useful properties that can be used for calculation of the initial probability assignment vector is the average gray level in a strip on each side of the segment. These averages can then be compared with typical gray levels of cultural features such as roads or buildings. The minimum difference of these side average gray levels from the typical gray levels of roads and buildings is used as a figure of merit in the calculation of initial probabilities.

Roads and buildings are the brightest objects on the photographs that we used. They also have similar gray levels (similar reflectances) in the scene. Using these facts, in what follows an automatic method for estimating the gray level is

*Permanent address: Shiraz University, College of Engineering, Shiraz, Iran.

described.

1) Calculate the average gray level in a strip on each side of each line segment.
2) Sort the line segments in decreasing order of length.
3) Select the longest p% of the lines (usually 5%).
4) Calculate the average gray levels of the brightest sides of the lines selected in step (3).

The average gray level calculated in this way can be accepted as a good estimate for the typical gray level of the objects.

To define the process of calculating the figures of merit more precisely, each line segment in the scene has two sides. The average gray levels of the strips along the two sides of the segment are denoted by g1 and g2 (see Figure 1).

Suppose that the typical average gray levels of roads and buildings are gr and gh respectively. Then the differences

$$f1 = |gr - g1| \quad \text{and} \quad f2 = |gr - g2|$$

measure the dissimilarity between the two sides of the line segment and the gray level of a typical road. Therefore, the function sr = min(f1,f2) is a measure of the gray level similarity between the given line segment and a typical road. Similarly the differences

$$h1 = |gh - g1| \quad \text{and} \quad h2 = |gh - g2|$$

measure the dissimilarity between the two sides of the line segment and the gray level of a typical building, and the function sh = min(h1,h2) is a measure of the gray level similarity between the given line segment and a typical building.

Finally,

$$s = \min(sr,sh)$$

will be small if the gray level average on one of the sides of the line segment is close to the gray level of a typical building or road. Therefore, if s is small the line segment is more probable to be an edge of a house or a building than to be an "other" type of edge, whereas if s has a large value, the probability that the line segment is in the "other" class is high.

In order to express the value of s as a figure of merit, linear functions are used. Let di (i = 1,2,3) represent the figures of merit. To define them as linear functions of s, the following linear expression is used for calculation of a road figure of merit. This linear function is shown in Figure 2 by thin solid lines.

$$d1 = \begin{cases} (1/dgr - 1/gr)(g-gr) + 1 \\ \quad \text{when } (g^2r - 2gr\ dgr)/(gr-dgr) \le g \le gr \\ 0 \quad \text{when } g^2r/(gr-dgr) < g < (g^2r-2gr\ dgr)/(gr-dgr) \\ (1/gr - 1/dgr)(g-gr) + 1 \\ \quad \text{when } gr \le g \le g^2r/(gr-dgr) \end{cases}$$

Here dgr is the deviation allowed for road gray level; beyond it, the figure of merit of "other" will become greater than the figure of merit of road. The value of g is

$$g = g1 \quad \text{if } f1 < f2$$

and

$$g = g2 \quad \text{if } f1 > f2$$

Similarly the figure of merit for a line segment being a piece of a building is shown by the thick solid lines in Figure 2 and its expression is as follows:

$$d2 = \begin{cases} (1/dgh - 1/gh)(g-gh) + 1 \\ \quad \text{when } g^2h - 2gh\ dgh)/(gh-dgh) \le g \le gh \\ 0 \quad \text{when } g^2h/(gh-dgh) < g < (g^2h-2gh\ dgh)/(gh-dgh) \\ (1/gh - 1/dgh)(g-gh) + 1 \\ \quad \text{when } gh \le g \le g^2h/(gh-dgh) \end{cases}$$

Here dgh is the deviation allowed for building gray level; beyond it, the figure of merit of "other" becomes greater than the figure of merit of buildings. The value of g is

$$g = g1 \quad \text{if } h1 < h2$$

and

$$g = g2 \quad \text{if } h1 > h2$$

When sr < sh road is more probable; therefore we use the dashed line for calculation of the figure of merit for "other". Similarly when sr > sh buildings are more probable and the dotted line is used for calculation of the figure of merit for "other". In summary, the figure of merit for the "other" class is calculated using the following formula:

When sr < sh

$$d3 = \begin{cases} |gr - g|/gr & \text{when } 0 < g < 2gr \\ 1 & \text{when } g \ge 2gr \end{cases}$$

Similarly when sr > sh

$$d3 = \begin{cases} |gh - g|/gh & \text{when } 0 < g < 2gh \\ 1 & \text{when } g \ge 2gh \end{cases}$$

The initial probability for each label is obtained by dividing the figure of merit of each label by the sum of the figures of merit of the three labels. Defining the initial probability in this manner, we have

$$p_\lambda^{(0)}(i) = di / \sum_{i=1}^{3} di \qquad i = 1,2,3$$

where di (i = 1,2,3) is the figure of merit of each label using the previous linear formulation and λ is the edge segment label.

When we use the functions in Figure 2, many segments will have probability 1 of belonging to the "other" class. These segments can be discarded as noise.

CALCULATION OF AVERAGE GRAY LEVEL ON BOTH SIDES OF AN EDGE

In order to calculate the initial probability assignments, we have to find the average gray level on both sides of a line. The algorithm for calculation of average gray level on both sides of an edge segment is as follows:

1) Generate a strip of width "d" on each side of the segment. Find the co-ordinates of the points inside the two strips as well as the number of points on each side.

2) Calculate the average gray level on each side by dividing the sum of the gray levels by the number of points on each side.

The algorithm starts by reading in the co-ordinates of the end points of each line. Then the slope of the line is calculated. At this point it is determined whether the angle (θ) of the line with respect to the x-axis is between 0 and 90 degrees or is between 90 and 180 degrees. This differentiation is necessary in order to define a sense for each side of the line segment.

Referring to Figure 3, the end points are designated as end point 1 and end point 2. The sides are denoted similarly. Using the conventions in Figure 3, the following equations can be written for each edge segment and for the boundaries of the strips on both sides of each segment. When θ is not equal to 90 degrees we have:

$$y_0(x) = mx + mx_1 + y_1$$
$$y_{13}(x) = -x/m + x_1/m + y_1$$
$$y_{14}(x) = -x/m + x_2/m + y_2$$
$$y_{11}(x) = mx - m(x_1 + \Delta x) + y_1 - \Delta y$$
$$y_{12}(x) = mx - m(x_1 - \Delta x) + y_1 + \Delta y$$

where $\Delta x = d \sin\theta$, $m = (y_1 - y_2)/(x_1 - x_2)$

$\Delta y = d \cos\theta$ when $0 \le \theta < 90$

and $\Delta y = -d \cos\theta$ when $90 < \theta < 180$

When θ is equal to 90 degrees we have the following equations for the boundary lines of the strip. This case is shown in Figure 4 and the equations are:

$$x_0 = x_1 = x_2, \qquad x_{11} = x_0 + d$$
$$x_{12} = x_0 - d, \qquad y_{13} = y_1$$
$$y_{14} = y_2$$

The digitized image is given in the form of a rectangular matrix of elements g(i,j) in which (i,j) are the Cartesian coordinates of a point and g(i,j) is the value of the brightness at the point (i,j).

In order to calculate the gray level averages inside the strips, we sum up the gray levels of those points which satisfy the conditions below and divide by the number of points in the strip:

$$\text{Average gray level} = \sum_{i,j} g(i,j)/n$$

The points inside each strip should satisfy the following conditions:

1) When $0° \le \theta < 90°$

   a) For side "1"

   $x_2 \le i \le x_1 + \Delta x$ $\qquad$ $y_2 - \Delta y \le j \le y_1$

   $y_{11}(i) < j < y_0(i)$ $\qquad$ $y_{14}(i) < j < y_{13}(i)$

   b) For side "2"

   $x_2 - \Delta x \le i \le x_1$ $\qquad$ $y_2(i) \le j < y_1 + \Delta y$

   $y_0(i) < j < y_{12}(i)$ $\qquad$ $y_{14}(i) < j < y_{13}(i)$

2) When $90° < \theta < 180°$

   a) For side "1"

   $x_1 \le i \le x_2 + \Delta x$ $\qquad$ $y_2 \le j \le y_1 + \Delta y$

   $y_0(i) < j < y_{11}(i)$ $\qquad$ $y_{14}(i) < j < y_{13}(i)$

   b) For side "2"

   $x_1 - \Delta x \le i \le x_2$ $\qquad$ $y_2 - \Delta y \le j \le y_1$

   $y_{12}(i) < j < y_0(i)$ $\qquad$ $y_{14}(i) < j < y_{13}(i)$

3) When $\theta = 90$

   a) For side "1"

   $x_1 - d \le i < x_i$ $\qquad$ $y_1 \le j \le y_2$

   b) For side "2"

   $x_1 < 1 < x_1 + d$ $\qquad$ $y_1 \le j \le y_2$

FINDING PAIRS OF COMPATIBLE SEGMENTS

The next step after noise cleaning is to group the line segments in a meaningful manner. In order to do this, models of the edges constituting objects should be used. The models of roads and buildings used in the program will now be described.

a) The model of edges belonging to a piece of a road

From the function of a road, it follows that certain physical and geometrical requirements must be satisfied. The properties used in this model are as follows:

1) The spectral properties of a road correspond to materials such as concrete and asphalt and it is usually homogeneous.

2) A piece of an edge of a road should have an anti-parallel edge.

3) A piece of an edge of a road is usually connected to other neighboring pieces with low angle deviation.

b) The model of edges belonging to a building

Similarly, the physical and geometrical properties of a building are:

1) The spectral properties of the roof of the building.

2) The similarity of gray level inside the edges constituting a building.

3) A piece of an edge of a building is connected to other pieces.

4) The edges of a building form a closed figure (usually with right angles).

In order to use the above models the geometric relationships between each pair of lines within a neighborhood in the scene should be studied. In general, using the conventions of Figure 3, every pair of lines in the scene belongs to one of sixteen cases. These cases are listed in Table 1. The entry "side" in Table 1 refers to the object side of the given segment.

In order to find the object side of a line segment, first the two values sr and sh are calculated. Then, using the following decision rules the object side is found:

```
when sr < sh
  if f1 < f2        side = 1
else                side = 2
and
when sr > sh
  if h1 < h2        side = 1
else                side = 2
```

Assume that the pair of lines under study are labeled as line A and line B. The angles of the two lines with respect to the x-axis are θ_A and θ_B respectively. Depending on the orientation of the pair of lines, different angles between the two lines are possible. Figure 5 shows examples of the angle θ between two lines. The plus sign indicates the side of the road or building. According to this convention the angle between two collinear lines is 180°.

Referring to the model of edges constituting the objects, each of these pairs of lines should satisfy certain conditions in order to be accepted as a candidate compatible pair. In general, these conditions are:

a) Similarity of gray level of a strip along a line connecting their ends with respect to the object side of the pairs.

b) Conditions on the geometrical configuration of the pair of lines.

To check the similarity condition, the average gray level on the object side of the pair of lines is calculated by

$$g = (gA + gB)/2$$

where gA and gB are the average gray levels of the strips along the object sides of lines A and B. Then, the corresponding average gray level of a strip along a line connecting the ends of the lines is calculated. The difference between this value and g is a measure of the gray level similarity of the line connecting the two ends with the pairs of lines. If this difference is within the limits used in calculation of the figures of merit, then the similarity condition is satisfied.

In a case where the distance between the ends is very small, that is, comparable with the width of the strip used in calculation of the gray level, the similarity measure is not reliable. This is because the number of points used in calculation of the average gray level is limited. In cases where the distance between the ends of pair under study is less than the width of the strip used in calculation of the average gray level, the similarity condition will not be checked. In this case the pair is considered as a compatible candidate if the appropriate geometrical conditions are satisfied.

Geometrical conditions are important in making two lines compatible. Figure 6 and Figure 7 show examples of geometrically compatible and incompatible pairs, respectively. To differentiate between geometrically compatible and incompatible pairs, certain constraints on the geometrical locations of the end points are necessary. The ratio of the distances between end points can be used to reject the geometrically incompatible pairs.

In what follows, the first four cases in Table 1 will be analyzed and their compatibility conditions derived. The other cases have similar conditions.

Case (1)

Referring to Figure 8, there are five different configurations. In this case the compatibility of line A with respect to line B at end (2) or the compatibility of line B with respect to line A at end (1) is considered. Table 2 summarizes the conditions imposed in these cases. The parameter m in the table is taken to be 1.5. This allows some overlap between the pairs of compatible line segments. The angle between the two lines is

$$\theta = \pi + |\theta_A - \theta_B| \quad \text{if } \theta_B \leq \theta_A$$

and

$$\theta = \pi - |\theta_A - \theta_B| \quad \text{if } \theta_B \geq \theta_A$$

Case (2)

In this case seven different configurations are considered. These are shown in Figure 9. The compatibility of end (1) of line A or line B is considered. Table 3 summarizes the required

conditions. The angle between the two lines in this case is

$$\theta = 2\pi - |\theta_A - \theta_B| \quad \text{when } ya2 < y0$$

and

$$\theta = |\theta_A - \theta_B| \quad \text{when } ya2 > y0$$

where

$$y_0 = mb\ xa2 - mb\ xb1 + yb1$$

and mb is the slope of line B. The conditions at end (2) of the lines are similar to the end (1) conditions. To find these conditions a1 and b1 should be changed to a2 and b2 except that in this case

$$y_0 = mb\ xa1 - mb\ xb1 + yb1$$

The side similarity for some configurations is different in this case.

Case (3)

When the compatibility of end (1) of line A with end (2) of line B is considered, there are five different configurations. Figure 10 shows these configurations. The conditions are summarized in Table 4. The angle between the lines is

$$\theta = \pi + |\theta_A - \theta_B|$$

The other possibility is to study the compatibility of end (2) of line A with end (1) of line B. Here again there are five different configurations. Figure 11 shows these configurations. The conditions are summarized in Table 5. The angle between the lines is

$$\theta = \pi - |\theta_A - \theta_B|.$$

Case (4)

The conditions for this case are summarized in Table 6 and Table 7. The different configurations are shown in Figure 12 and Figure 13. The angle between the lines is

$$\theta = |\theta_A - \theta_B|$$

when the compatibility of end (1) of line A is considered. Similarly the angle is

$$\theta = |\theta_A - \theta_B|$$

when the compatibility of end (2) of line A is in question.

So far the geometrical and similarity conditions for the pairs of compatible pieces have been found. In what follows the algorithm for finding compatible pairs will be explained.

Algorithm for Finding Compatible Pairs

1) Choose those line segments whose "other" probability is not equal to 1.

2) For end "1" of each line, find the shortest distances from other end points of line segments.

3) Find the object side of the given line and the other lines found in (2).

4) Check the geometrical and similarity conditions for the given line and the other lines found in (2). Reject those lines for which the required conditions are not satisfied.

5) If all the lines are rejected go to (8).

6) Find the angle of the line with respect to the remaining lines in (4). Choose the line which has the smallest angle (e.g. greater than 25°) with respect to the line under study.

7) Choose the other end of the line found in (6) and go to (2).

8) Choose the other end of the given line and go to (2). If the other end has already been tested go to (9).

9) Continue the above process for the other line segments.

FINDING PAIRS OF ANTIPARALLEL EDGES

The edges of cultural features usually occur in pairs, as in the sides of roads and of buildings. To identify these features the edges should be clustered into antiparallel pairs (i.e. pairs of facing edges that are parallel but have opposite senses). Clustering must take into account information from the picture in the regions around the edges. For example, a road usually has a uniform gray level and thus it is reasonable to expect the facing sides of an antiparallel pair of edges to have similar gray levels. Previous work [4,7] has restricted the choice of pairs to lines that are closest neighbors. A method of pairing antiparallel straight lines reported in [8] is based on the distance between the lines, the amount by which they overlap, and on whether or not other lines are interposed.

The present method finds the pairs of lines that are anti-parallel up to a certain angle difference (usually 25°) when similarity of gray level between the pairs is satisfied.

The basic procedure is as follows. A strip is moved along the object side of each edge segment. The movement is continued until the similarity is lost or the distance moved is greater than the largest expected object size in the scene. While the strip moves, it hits other line segments. Among these line segments the following segments are rejected:

a) If they are not anti-parallel

b) If the difference in the angle is greater than a threshold.

The similarity is defined as the difference between the average gray level of the moving strip and the average gray level of the object side of the edge segment:

$$|g - gmove| < \text{level of similarity}$$

where g = average gray level of the line segment
and gmove = average gray level of the moving strip.

The level of similarity used in the program is taken as 7, which is a rather tolerant condition. When the strip hits a candidate line the level of similarity is automatically changed to the value of the contrast of the candidate line. Note that this change of the level may stop the movement of the strip.

Among the remaining lines the one which has the smallest distance is selected as anti-parallel. To find the shortest distance between two anti-parallel line segments, at each end of the two segments perpendicular lines are drawn to the other line. If the intersection of the perpendicular with the facing line is located outside of the line, the distance is neglected. Among the remaining distances, the minimum is selected as the distance between the two lines. Figure 14 shows an example of calculating the distance between two lines. As shown in this figure, among the four distances

$$di \quad (i = 1,2,3,4)$$

d3 and d4 are rejected and

$$d = \min(d1,d2)$$

is selected as the distance between the two lines.

To check whether the intersection of the perpendicular line is between the end points of a line the following decision rules are used:

if θ is not equal to 90 degrees
and $x1 \leq xint \leq x2$ the intersect point is between the end points

else if θ is equal to 90 degrees
and $y1 \leq yint \leq y2$ the intersect point is between the end points.

Here (xint,yint) are the coordinates of the intersection point.

This method of finding anti-parallel pairs of lines has the following advantages:

a) Each line is not compared with all other lines.

b) When several lines are facing a line, the method allows all of these lines to choose the same line as anti-parallel.

c) The method uses the context of the lines on the picture, namely, the similarity of the gray levels inside the object.

In what follows the algorithm for finding anti-parallel pairs is explained.

Algorithm for finding anti-parallel pairs

1) Choose a line segment and find its object side.

2) Generate the strip (a width of 4 points is used), and find the average gray level inside the strip.

3) Move this strip parallel to the segment. While the similarity and the total moving distance are less than the specified levels, note the lines hit by the strip. If no lines are found go to (5). Otherwise, reject those lines where the angle difference is greater than the specified threshold and the facing side is not opposite to the original line. Set the similarity level equal to the contrast of the line found and continue the process.

4) For the candidate lines found in (3) choose the one which has the minimum distance. Mark the line found in order not to process it again.

5) Continue the process for the remaining lines.

The maximum moving distance in the above algorithm is quite relaxed; it is set to be equal to be 1/4 of the size of the picture. For scenes containing small objects this distance can be reduced in order to reduce computation time. The angle difference can be set arbitrarily. The program is not sensitive to this threshold since the strip move along the object side of the edge and so it is expected that we get another side of the object as the best candidate.

STUDY OF SHADOWS OF BUILDINGS

One of the features that can be used for verification of the recognition of buildings is the shadow of the building. There are cases where a parking lot can be recognized as a building, since they may have the same size or shape. It seems that it is possible to use the shadow for further verification. To study this,some experiments have been performed using the average gray level within a strip along each line segment and the angle of the line segment.

Referring to Figure 15 each line segment with angle θ with respect to the x-axis has two sides. There are two average gray levels associated with each line segment. The average gray levels g1 and g2 are associated with angles θ and $2\pi-\theta$ respectively.

Scatter plots of g1 and g2 with respect to θ are shown in Figure 16 for all of the line segments and in Figure 17 for the line segments after noise cleaning. Figure 17 shows that at dark gray levels, the population of points for $\theta \geq 180°$ is greater than the population of points for $\theta < 180°$. This shows that in certain orientations along the line segments, there exist dark shadow regions.

To study this effect quantitatively, let us pick the darkest P% of the population. Let

n1 = number of dark points in the interval $(\theta, \theta+\pi)$

and

n2 = number of dark points in the interval $(\theta+\pi, \theta+2\pi)$

for $\theta = 0, 20, 40, \ldots, 340$. The plots of n1/n2 as a function of θ for different values of P and for the line segments after and before noise cleaning are shown in Figure 18 and Figure 19 respectively. These figures show that there is a peak around 180°. The peak is greater for the segments after noise removal. As expected, if P is increased the peak becomes smaller. This effect was tested on several other scenes and similar results were obtained.

THE RECOGNITION PROCESS

After application of the programs described up to now, we have groups of compatible segments (also the angles between them) and pairs of anti-parallel segments. Using the model of roads and buildings, we want to update the probabilities that were initially obtained using gray level information. Based on these probabilities we can recognize objects with good confidence or fair confidence.

We begin by dividng the groups of compatible pairs into the following categories:

A) Closed groups

B) Semiclosed groups

C) Other lines and groups

In what follows each of the above categories will be explained in more detail.

A) Closed groups

By a closed group, we mean that the start and the end segment labels are the same. Figure 20 shows an example of this type of group. In this figure A,B,C.... are the labels in a compatible group.

It is obvious that this kind of closed group is a good candidate for being the group of edges of a building. To check whether this closed group is a house, we test for solidness inside the object sides, and also check that each line segment in the group is antiparallel to a line in the group. To check solidness we use the same operator that was used in finding the antiparallel pairs. This test also guarantees the similarity of gray level inside the object.

The above check can differentiate between the cases (b) and (c) in Figure 20. Thus a closed group with the above conditions can be considered a house with good confidence. Furthermore the shadow can be used for further verification.

B) Semiclosed groups

A semiclosed group is defined as a group with a gap less than the longest line connecting the ends of compatible pairs in the group. Figure 21 demonstrates an example of this type. As in the case of a closed group, if the following tests are valid, then the group is accepted as a house with good confidence.

1) Solidness

2) Each line should be antiparallel to a line in the group.

Operators similar to those used for checking closed groups are used here. The shadow can be used for further verification.

C) Other lines and groups

Here again the model of the edges constituting a house or road will be used for the recognition of the remaining lines or groups. The important features are the angles between the compatible pairs and information on anti-parallel pairs. Figure 22 shows examples of possible cases that may occur in the scene. In this Figure θmin is around 200°. Special care should be taken in cases where the anti-parallel pairs or compatible pairs are not available, due to cutoff at an edge of the frame. The implementation of this part is in progress.

EXAMPLES

In order to test the approach, a set of digitized images of a suburban area in Occoquan, VA were used. The straight edges in the images were extracted using the iterative enhancement technique of [6]. The resulting lines and the original gray scale picture were used as input to the program. Figure 23 shows one of the input images and Figure 24 shows the set of lines extracted from it. The data of Figure 24 (the end point coordinates) along with the gray scale input picture are the input to another program for calculation of the average gray levels on both sides of the line segments. The width of the strip is taken to be four points. This value was selected based on knowledge about the resolution of the picture. After calculation of the averages on both sides, the vector of figures of merit is calculated according to Section 2. The typical gray levels of roads and buildings are taken to be equal in this case. The histogram of the probability of "other" is shown in Figure 25. From this histogram it is clear that we can

completely differentiate between two classes of object boundaries, namely objects and noise. Figure 26 shows the line segments whose probabilities of being a piece of road or building are not equal to zero. This figure shows quite an improvement in rejecting the noise edges. Figure 27 shows the results of finding compatible segments, and Figure 28 shows the results of finding anti-parallel segments: the midpoints of the anti-parallel pairs are connected together. Figure 29 shows the good-confidence houses. Figures 30 to 35 and 36 to 41 show the results of the program on two other scenes. So far, only the recognition of houses with good confidence has been implemented. Work is in progress on other cases.

REFERENCES

1. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, 1979.
2. W. K. Pratt, Digital Image Processing, John Wiley & Sons, 1979.
3. L. S. Davis, A Survey of Edge Detection Techniques, Computer Graphics and Image Processing, Vol. 4, No. 3, Sept. 1975.
4. R. Nevatia and K. R. Babu, Linear Feature Extraction, USCIPI Report 840, 1978.
5. R. Bajcsy and M. Tavakoli, Computer Recognition of Roads from Satellite Pictures, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, No. 9, Sept. 1976.
6. S. Peleg and A. Rosenfeld, Straight edge enhancement and mapping, Computer Science TR-694, University of Maryland, College Park, MD, Sept. 1978.
7. R. Brooks, Global directed edge linking and ribbon finding, Proc. DARPA Image Understanding Workshop, Menlo Park, CA, April 1979, 72-78.
8. Ann Scher, Michael Shneier, and Azriel Rosenfeld, A method for finding pairs of anti-parallel straight lines, Computer Science TR-845, University of Maryland, College Park, MD, Dec. 1979.

# of cases	Segment A case	Segment A side	Segment B case	Segment B side	Similar to:
1	a	1	a	1	--
2	a	1	a	2	--
3	a	1	b	1	--
4	a	1	b	2	--
5	a	2	a	1	#2
6	a	2	a	2	#1
7	a	2	b	1	#4 rotated 180°
8	a	2	b	2	#3 rotated 180°
9	b	1	a	1	#3
10	b	1	a	2	#7
11	b	1	b	1	#1
12	b	1	b	2	#2 rotated 90°
13	b	2	a	1	#4
14	b	2	a	2	#8
15	b	2	b	1	#12
16	b	2	b	2	#6

Table 1. Different cases of pairs of line segments according to the conventions of Figure 3.

Case 1	Geometrical conditions	Similarity conditions for the line a_2b_1	
	$a_2b_1 < a\,b\,/m$ $a_2b_1 < a_1b_1/m$	If $a_2b_1 < d$ or $x_pb_1 < d$ or $x_pa_2 < d$	no check
	$\theta_A \neq \theta_B$		
(a)	$x_{a_2} \leq x_p < x_{a_1}$ $x_{b_2} < x_p \leq x_{b_1}$	no check	
(b)	$x_{a_2} < x_{b_1}$ $y_{a_2} \leq y_{b_1}$	check side 2	
(c)	$x_{a_2} > x_{b_1}$ $y_{a_2} \geq y_{b_1}$	check side 1	
(d)	$x_{a_2} < x_{b_1}$ $y_{a_2} > y_{b_1}$	check side 1	
(e)	$x_{a_2} \geq x_{b_1}$ $y_{a_2} < y_{b_1}$	check side 2	

Table 2. Geometrical and Similarity conditions for case 1.

	Geometrical conditions	Similarity conditions for line a_1b_1
Case 2	$a_1b_1 < a_2b_2/m$ $a_1b_1 < a_1b_2/m$ $a_1b_1 < a_2b_1/m$	If $a_1b_1 < d$ no check
(a)	$\theta_A \neq \theta_B$ $x_{a_2} < x_p \leq x_{a_1}$ $x_{b_2} < x_p \leq x_{b_1}$ $x_pa_1 < x_pa_2/m$, $x_pb_1 < x_pb_2/m$	no check
(b)	$y_{a_2} < y_0$ $x_{a_1} < x_{b_1}, y_{b_1} > y_{a_1}$	If $\theta_A \neq \theta_B$ and $x_p < d$ no check else check side 1
(c)	$y_{a_2} < y_0$ $x_{a_1} > x_{b_1}$, $y_{b_1} \leq y_{a_1}$	If $\theta_A \neq \theta_B$ and $x_pb_1 < d$ no check else check side 2
(d)	$y_{a_2} < y_0$ $x_{a_1} \geq x_{b_1}, y_{b_1} > y_{a_1}$	check side 1
(e)	$y_{a_2} > y_0$ $x_{a_1} > x_{b_1}, y_{a_1} > y_{b_1}$	If $\theta_A \neq \theta_B$ and $x_pb_1 < d$ no check else check side 2
(f)	$y_{a_2} > y_0$ $x_{a_1} < x_{b_1}, y_{a_1} \leq y_{b_1}$	If $\theta_A \neq \theta_B$ and $x_pa_1 < d$ no check else check side 1
(g)	$y_{a_2} > y_0$ $x_{a_1} \leq x_{b_1}, y_{a_1} > y_{b_1}$	check side 2

Table 3. Geometrical and similarity conditions for Case 2.

	Geometrical conditions	Similarity conditions for line a_2b_1
Case 3 end 2	$a_2b_1 < a_1b_2/m$ $a_2b_1 < a_2b_2/m$ $a_2b_1 < a_1b_1/m$	If $a_2b_1 < d$ no check
(a)	$x_pa_2 < x_pa_1/m$ $x_pb_1 < x_pb_2/m$ $x_{a_2} \leq x_p < x_{a_1}$ If $\theta_B \neq 90°$ $x_{b_1} \leq x_p < x_{b_2}$ else $y_{b_2} < y_p \leq y_{b_1}$	no check
(b)	$y_{b_1} \leq y_{a_2}$ $x_{a_2} > x_{b_1}$	If $x_pa_2 < d$ no check else check side 1
(c)	$y_{b_1} < y_{a_2}$ $x_{a_2} \leq x_{b_1}$	check side 1
(d)	$y_{a_2} \leq y_{b_1}$ $x_{a_2} < x_{b_1}$	If $x_pb_1 < d$ no check else check side 2
(e)	$y_{a_2} \leq y_{b_1}$ $x_{a_2} > x_{b_1}$	If $x_pa_2 < d$ no check else check side 2

Table 5. Geometrical and similarity conditions for case 3: end 2 of line A.

	Geometrical conditions	Similarity conditions for line a_1b_2
Case 3 end 1	$a_1b_2 < a_2b_1/m$ $a_1b_2 < a_2b_2/m$ $a_1b_2 < a_1b_1/m$	If $a_1b_2 < d$ no check
(a)	$x_{a_2} < x_p \leq x_{a_1}$ $x_pa_1 < x_pa_2/m$ $x_pb_2 < x_pb_1/m$ If $\theta_B \neq 90°$ $x_{b_1} < x_p \leq x_{b_2}$ else $y_{b_2} < y_p < y_{b_1}$	no check
(b)	$x_{a_1} \geq x_{b_2}$ $y_{a_1} < y_{b_2}$	check side 1
(c)	$x_{a_1} < x_{b_2}$ $y_{a_1} \leq y_{b_2}$	check side 1
(d)	$x_{a_1} < x_{b_2}$ $y_{a_1} > y_{b_2}$	If $x_pa_1 < d$ no check else check side 2
	$x_{a_1} > x_{b_2}$ $y_{a_1} \geq y_{b_2}$	If $x_pb_2 < d$ no check else check side 2

Table 4. Geometrical and similarity conditions for case 3: end 1 of line A.

	Geometrical conditions	Similarity conditions for line a_1b_1
Case 4 end 1	$a_1b_1 < a_2b_2/m$ $a_1b_1 < a_2b_1/m$ $a_1b_1 < a_1b_2/m$ $x_pa_1 < x_pa_2/m$ $x_pb_1 < x_pb_2/m$	If $a_1b_1 < d$ no check
(a)	$x_{a_2} < x_p \leq x_{a_1}$ If $\theta_B \neq 90°$ $x_{b_1} \leq x_p < x_{b_2}$ else $y_{b_2} < y_p < y_{b_1}$	no check
(b)	$x_{a_1} \leq x_{b_1}$ $y_{a_1} > y_{b_1}$	check side 2
(c)	$x_{a_1} < x_{b_1}$ $y_{a_1} \leq y_{b_1}$	check side 1
(d)	$x_{a_1} > x_{b_1}$ $y_{a_1} > y_{b_1}$	If $x_pb_1 < d$ no check else check side 2
(e)	$y_{a_1} < y_{b_1}$ $x_{a_1} \geq x_{b_1}$	If $x_pa_1 < d$ no check else check side 1

Table 6. Geometrical and similarity conditions for Case 4: end 1 of line A.

	Geometrical conditions	Similarity conditions for line a_2b_2
Case 4 end 2	$a_2b_2 < a_1b_1/m$ $a_2b_2 < a_2b_1/m$ $a_2b_2 < a_1b_2/m$ $x_pa_2 < x_pa_1/m$ $x_pb_2 < x_pb_1/m$	If $a_2b_2 < d$ no check
(a)	$x_pa_2 \leq x_p < x_{a_1}$ If $\theta_B \neq 90°$ $x_{b_1} < x_p \leq x_{b_2}$ else $y_{b_1} \leq y_p < y_{b_1}$	no check
(b)	$x_{a_2} > x_{b_2}$ $y_{a_2} \geq y_{b_2}$	If $x_pa_2 < d$ no check else check side 1
(c)	$x_{a_2} \leq x_{b_2}$ $y_{a_2} > y_{b_2}$	If $x_pa_2 < d$ no check else check side 1
(d)	$x_{a_2} \geq x_{b_2}$ $y_{a_2} < y_{b_2}$	If $x_pb_2 < d$ no check else check side 2
(e)	$x_{a_2} < x_{b_2}$ $y_{a_2} < y_{b_2}$	If $x_pb_2 < d$ no check else check side 2

Table 7. Geometrical and similarity conditions for Case 4: end 2 of line A.

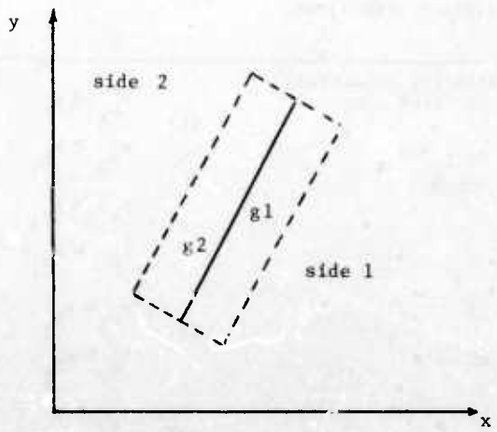


Figure 1. A line segment and strips along each side of it

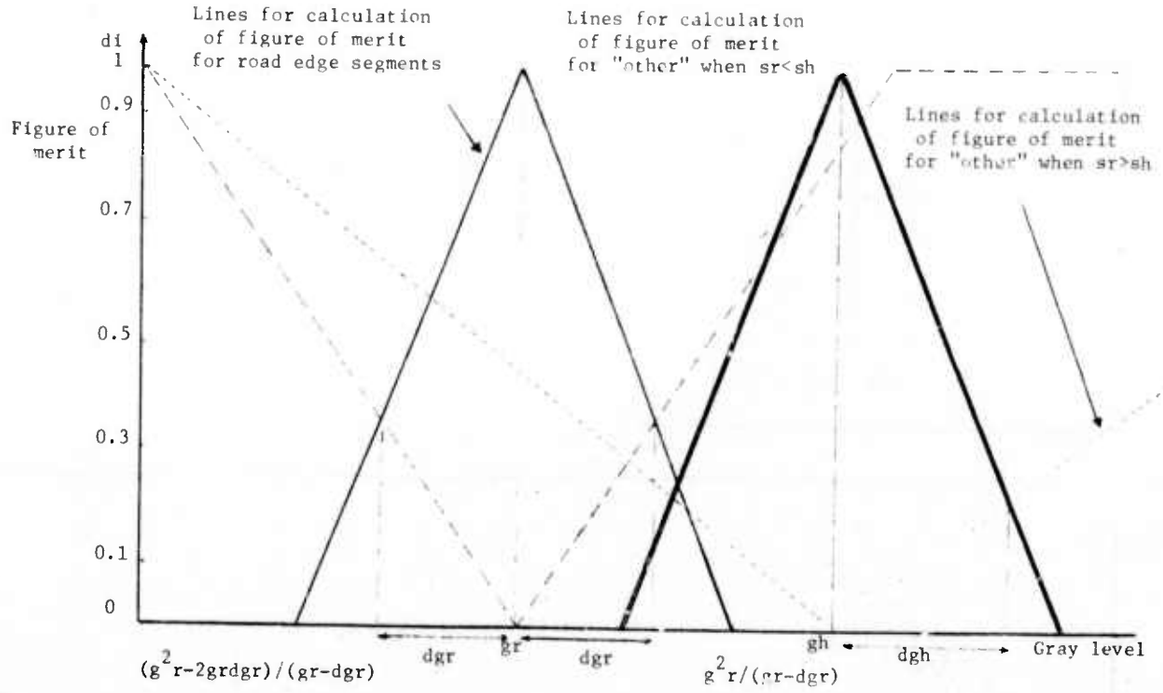


Figure 2. Linear functions for calculation of figures of merit

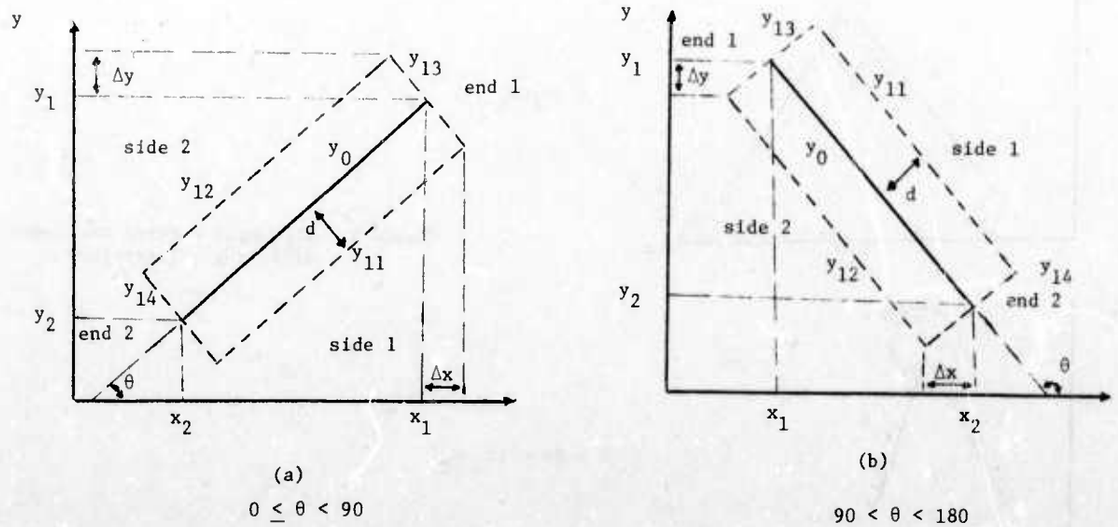


Figure 3. End and side conventions.

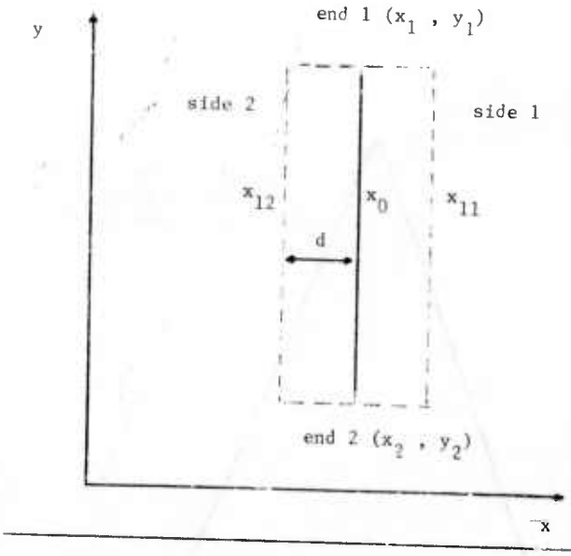
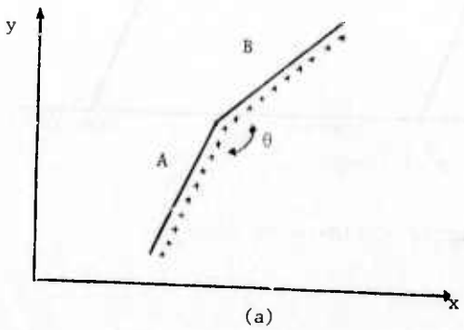


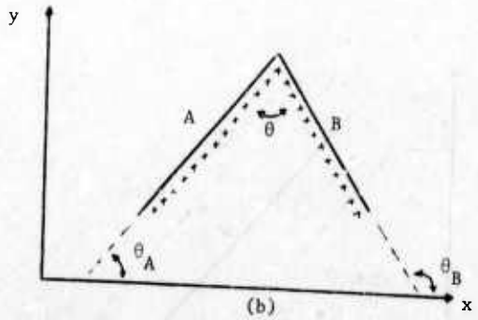
Figure 4. θ=90°



(a)

$$\theta = \pi + |\theta_A - \theta_B| \quad \text{if} \quad \theta_B \geq \theta_A$$

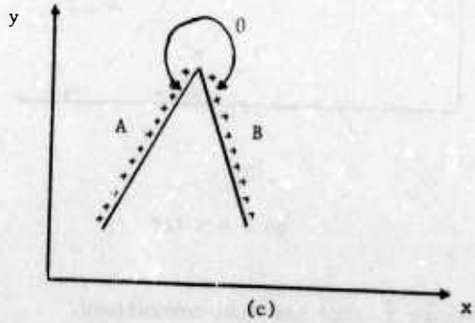
$$\theta = \pi - |\theta_A - \theta_B| \quad \text{if} \quad \theta_B \leq \theta_A$$



(b)

$$\theta = |\theta_A - \theta_B|$$

Figure 5. The angle between two lines in different orientations



(c)

$$\theta = 2\pi - |\theta_A - \theta_B|$$

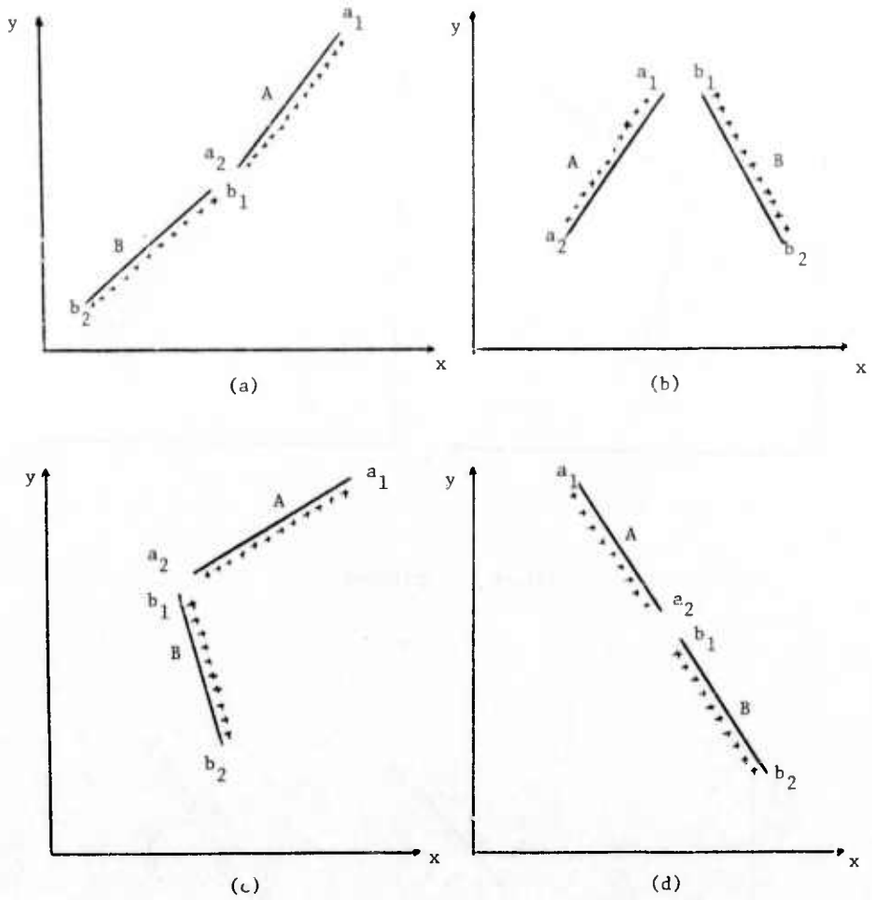


Figure 6. Examples of geometrically compatible candidate pairs.

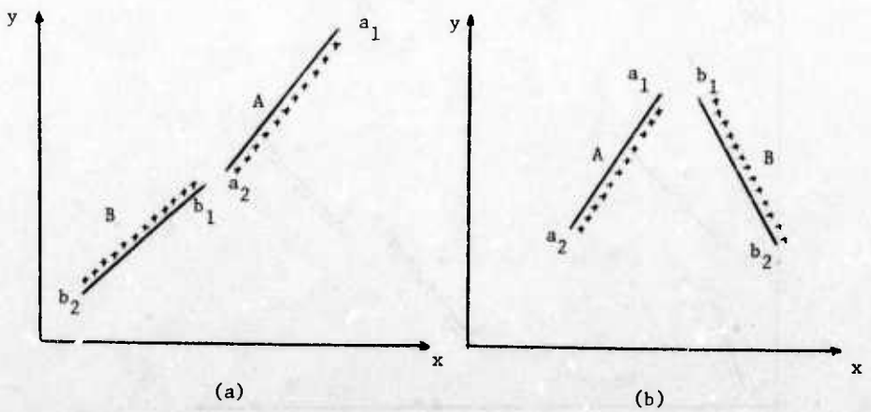


Figure 7. Examples of geometrically incompatible candidate pairs.

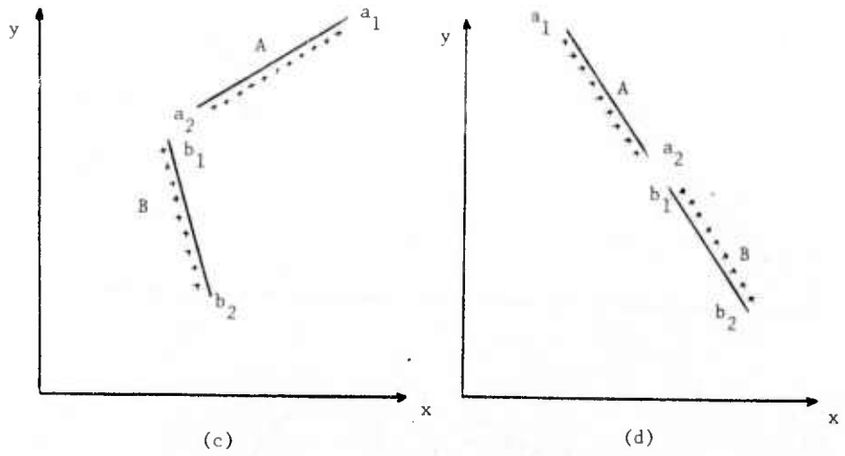


Figure 7, continued.

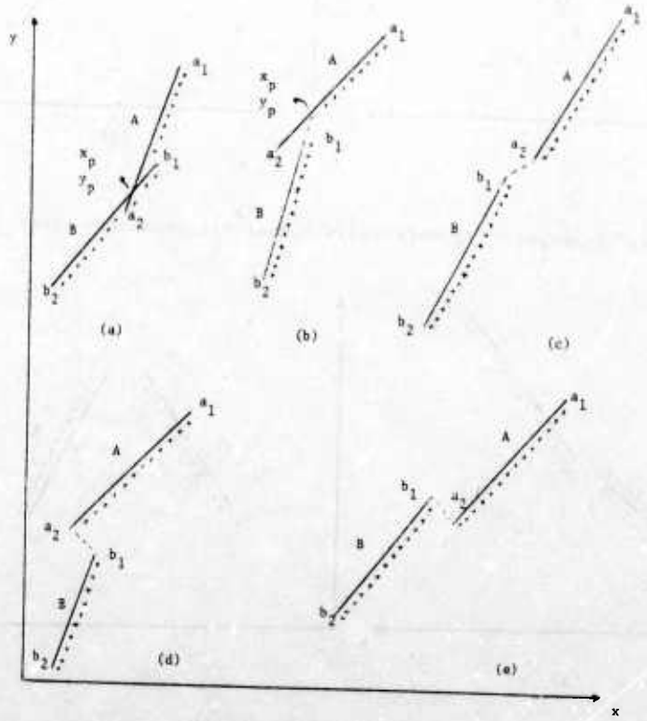


Figure 8. Configurations in Case 1

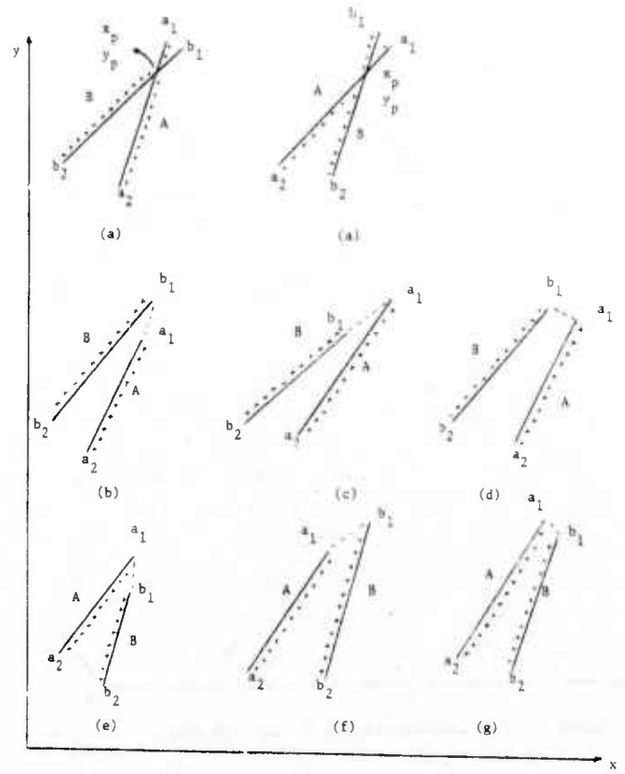


Figure 9. Configurations in Case 2

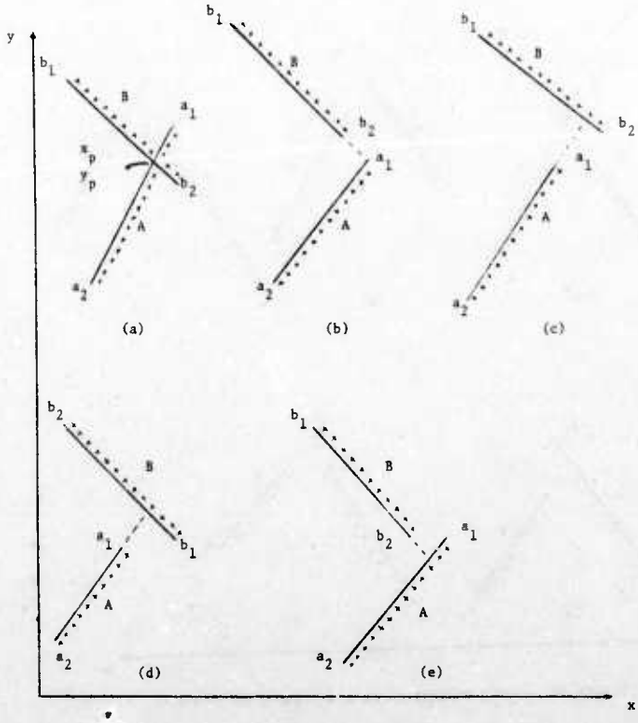


Figure 10. Configurations in Case 3: end 1 of line A

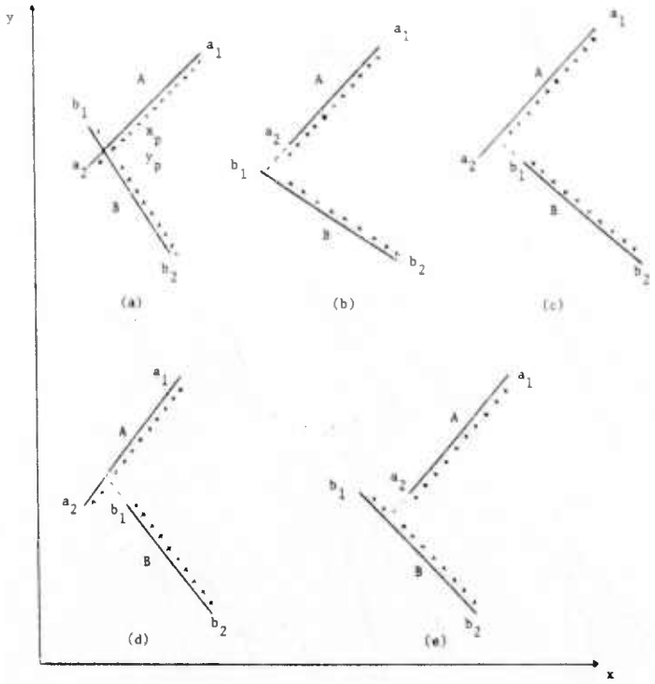


Figure 11. Configurations in Case 3: end 2 of line A

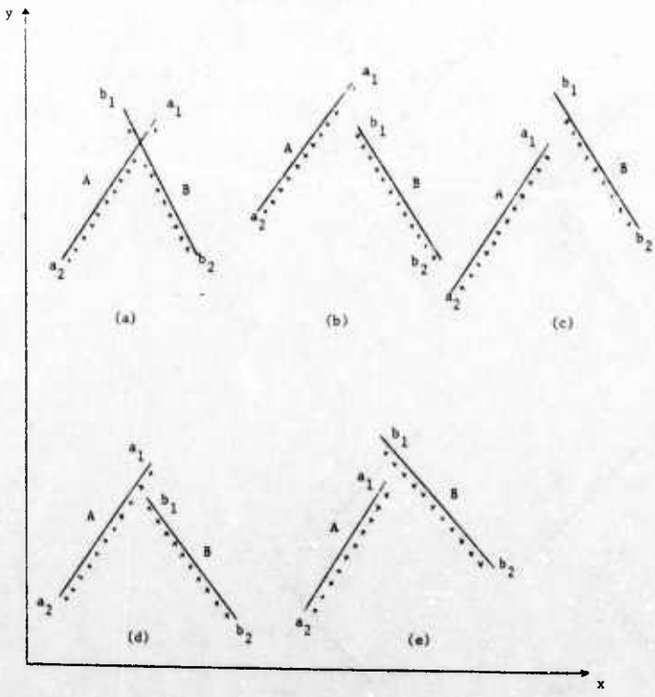


Figure 12. Configurations in Case 4: end 1 of line A

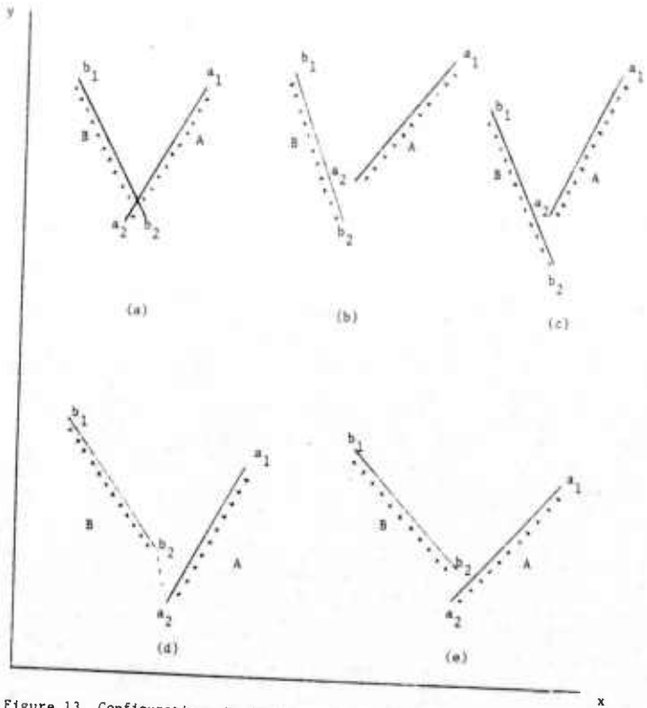


Figure 13. Configurations in Case 4: end 2 of line A.

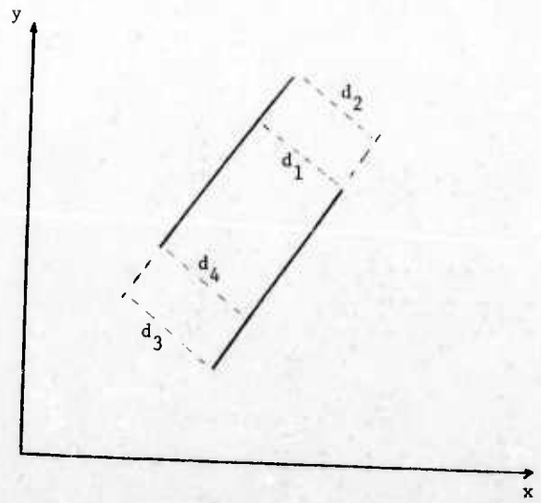


Figure 14. Calculation of the distances between two lines.

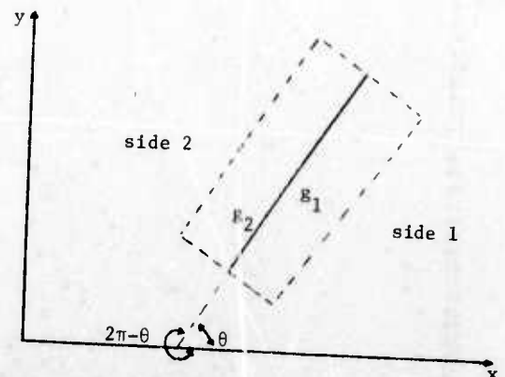


Figure 15. Angle convention for g_1 and g_2.

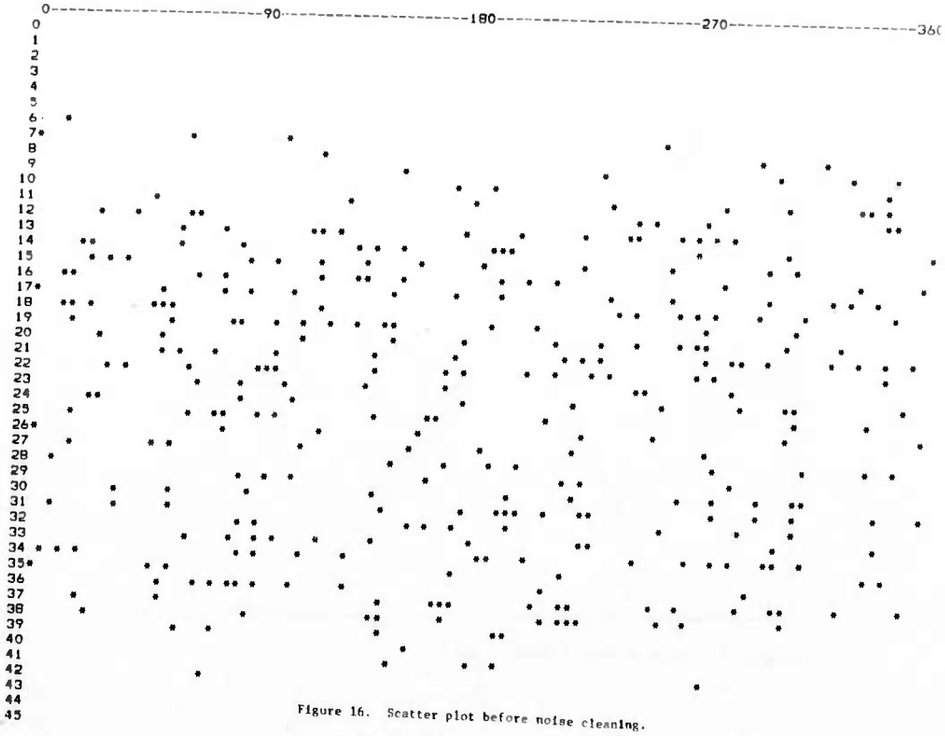


Figure 16. Scatter plot before noise cleaning.

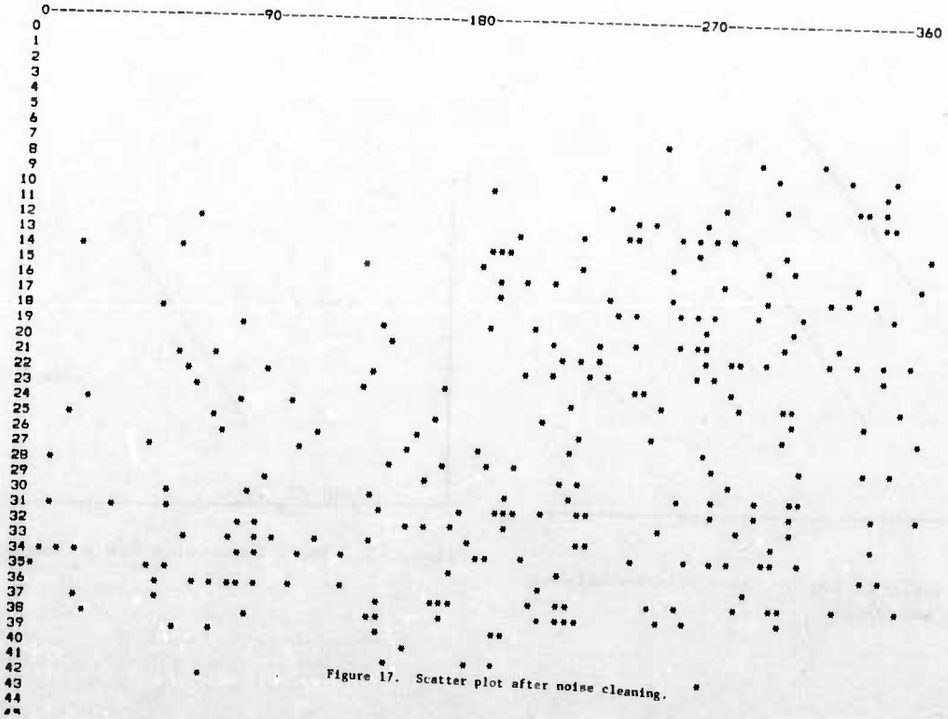


Figure 17. Scatter plot after noise cleaning.

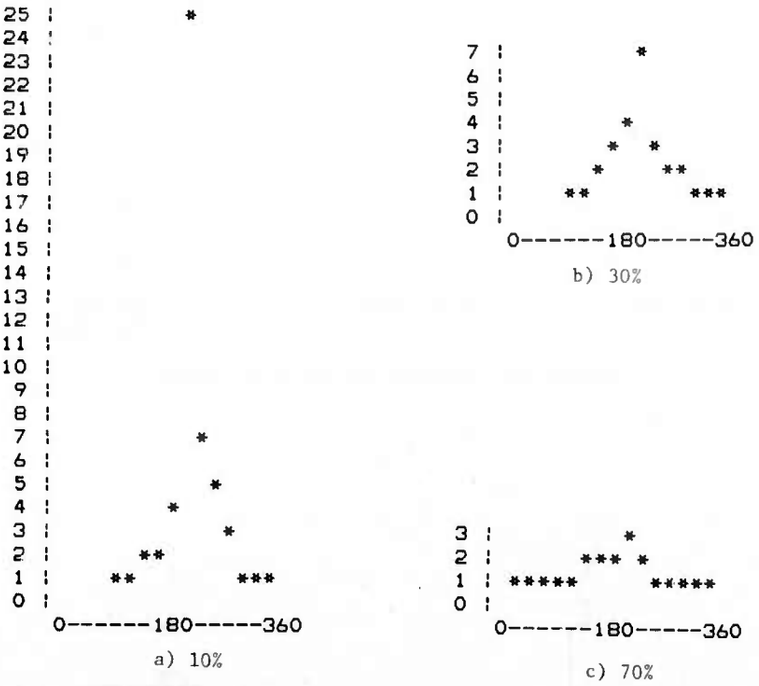


Figure 18. Plot of n_1/n_2 as a function of θ, after noise cleaning.

Figure 19. Plot of n_1/n_2 as a function of θ, before noise cleaning.

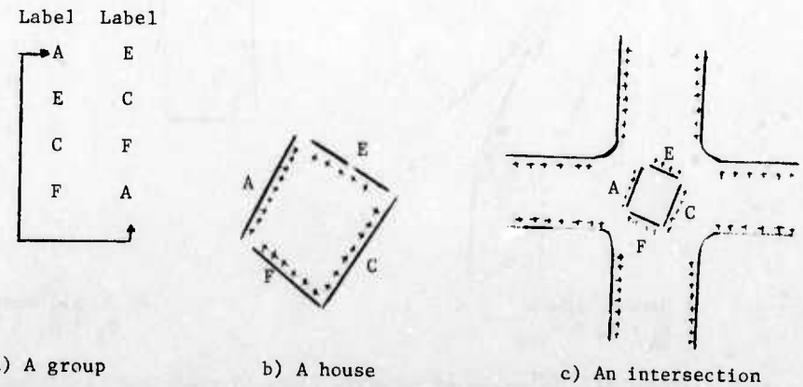


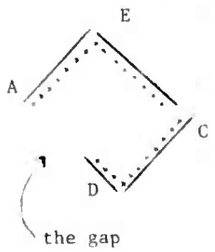
Figure 20. Examples of closed groups.

Label	Label
A	E
E	C
C	D

a) A group

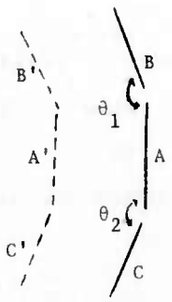
Label	Label
E	C
C	D
E	A

b) A group

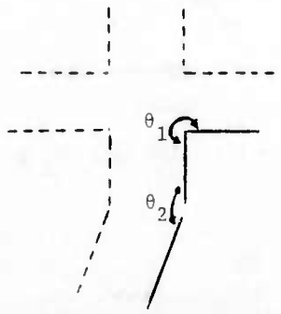


c) A house with missing edge.

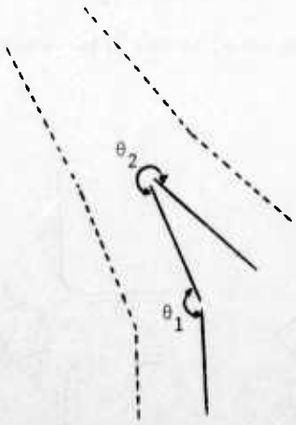
Figure 21. Examples of semiclosed groups.



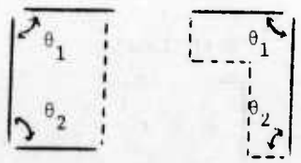
a) A curved road
$\theta_1 + \theta_2 > \theta_{min}$



b) Three or four way intersection
$\theta_1 + \theta_2 > \theta_{min}$



c) Merging roads
$\theta_1 + \theta_2 > \theta_{min}$



d) Single houses
$\theta_1 + \theta_2 < \theta_{min}$

Figure 22. Examples of possible cases of roads and buildings.

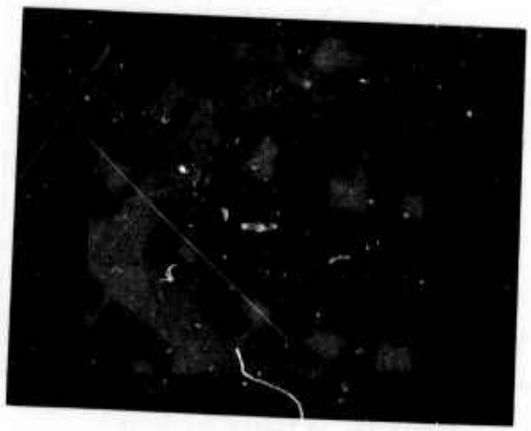


Figure 23. A suburban scene.

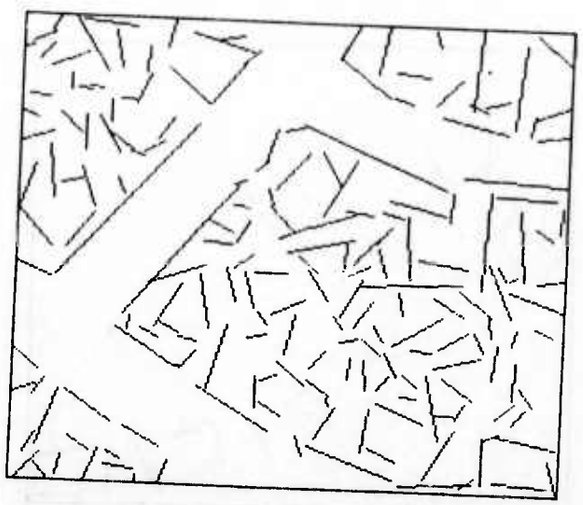


Figure 24. Line segments fitted to the edge components of the scene of Figure 23.

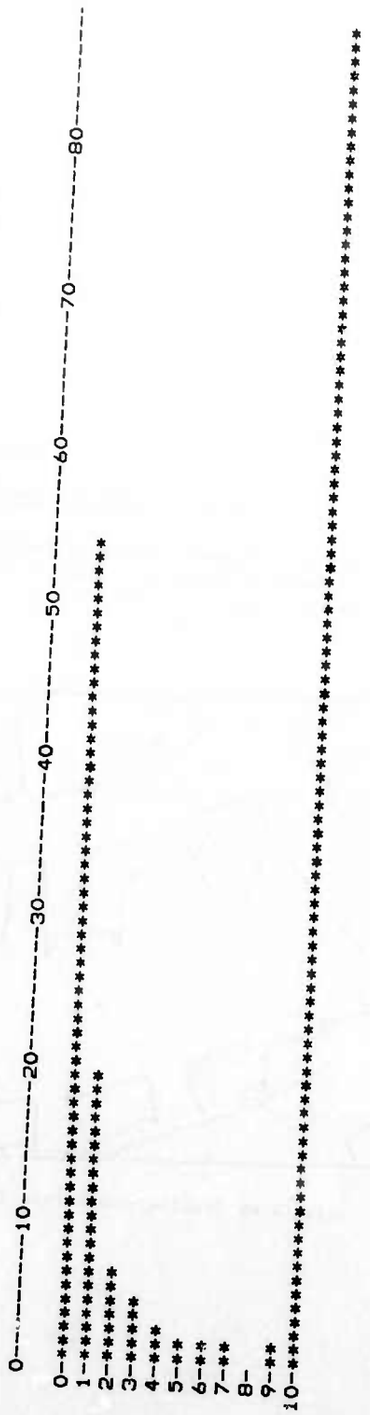


Figure 25. Histogram of the probability of "other".

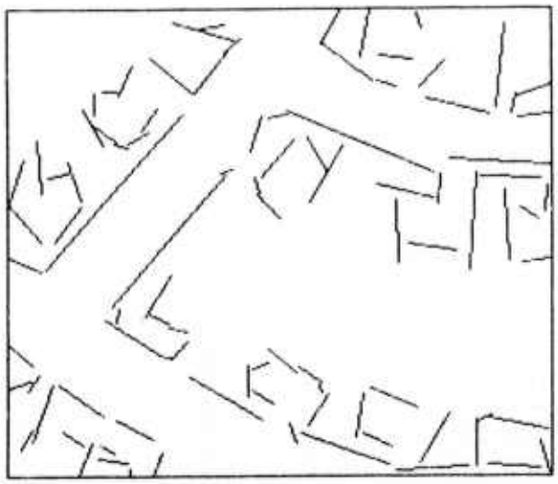


Figure 26. The line segments whose probabilities of being a piece of a road or building are not equal to zero.

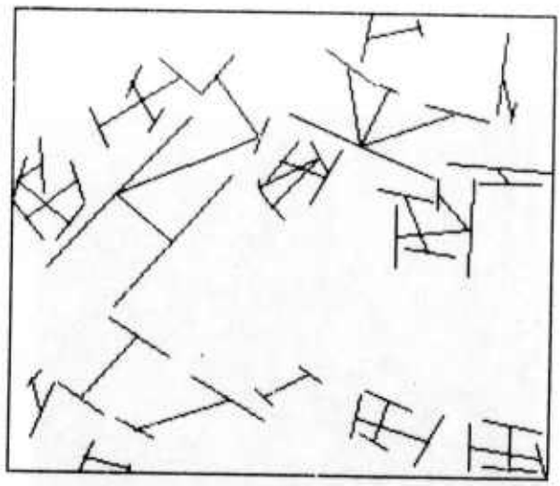


Figure 28. Antiparallel lines.

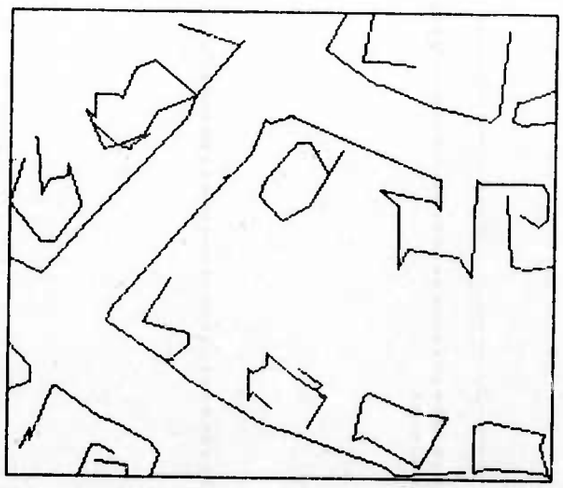


Figure 27. Results of linking compatible lines.

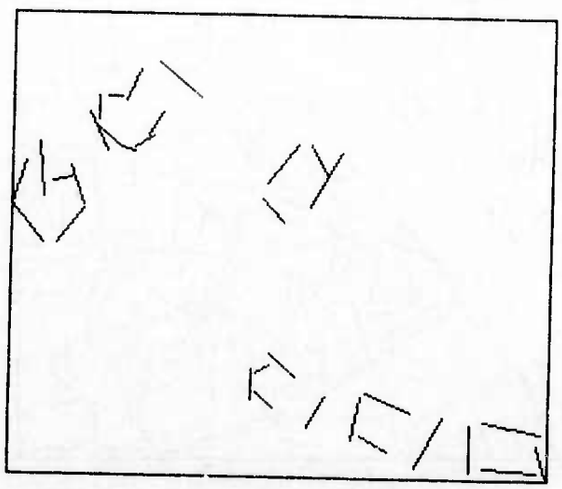


Figure 29. Houses with good confidence.

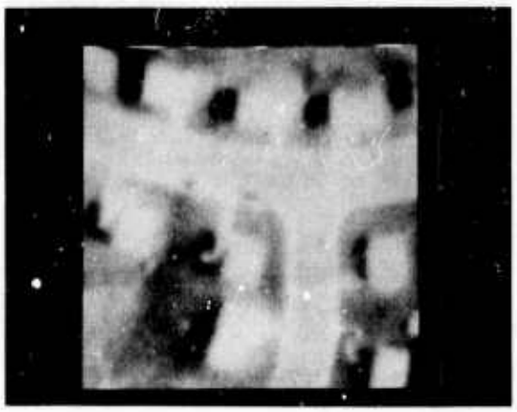


Figure 30. Another suburban scene

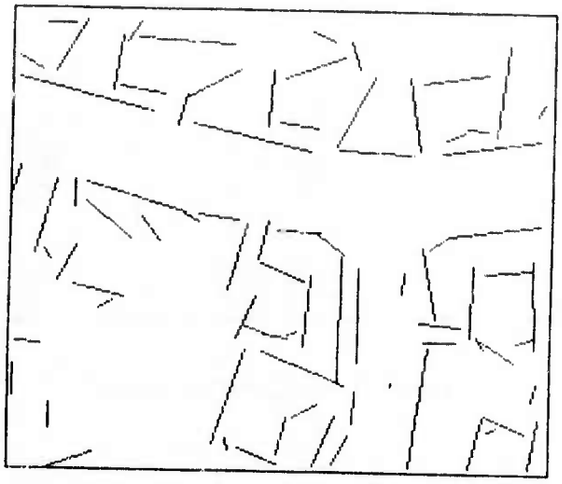


Figure 32. Line segments whose probabilities of being a piece of road or building are nonzero.

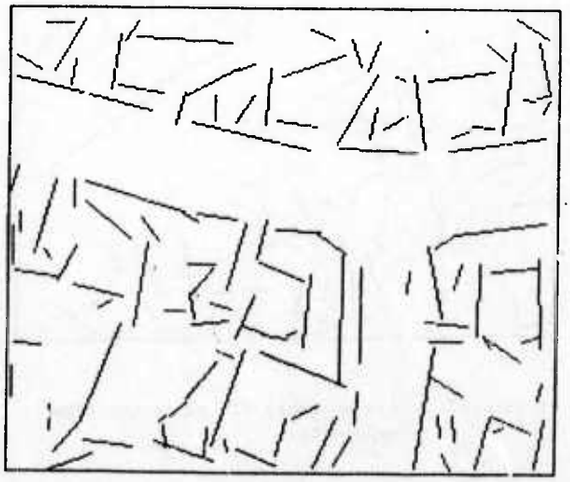


Figure 31. Line segments fitted to the edge components

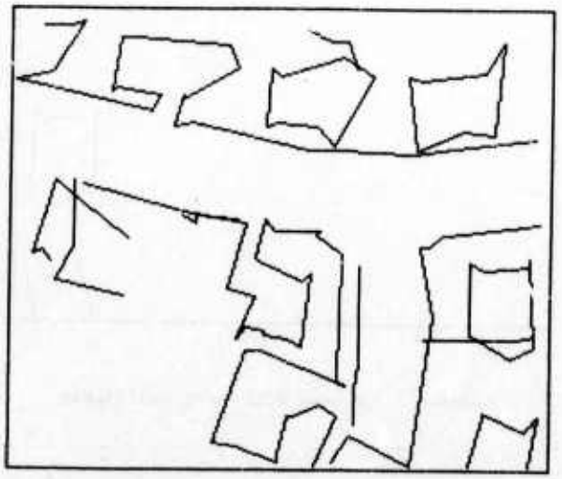


Figure 33. Compatible lines.

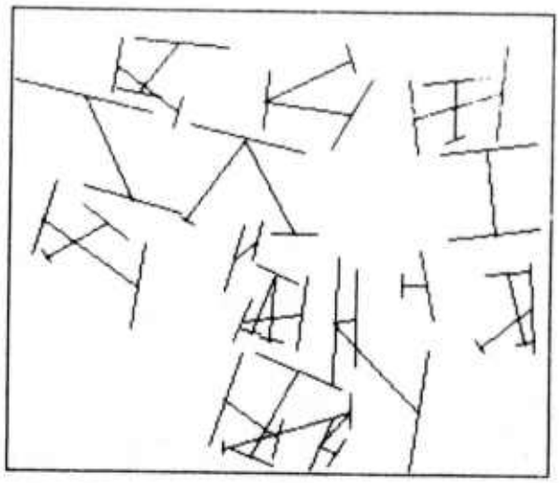


Figure 34. Antiparallel lines.

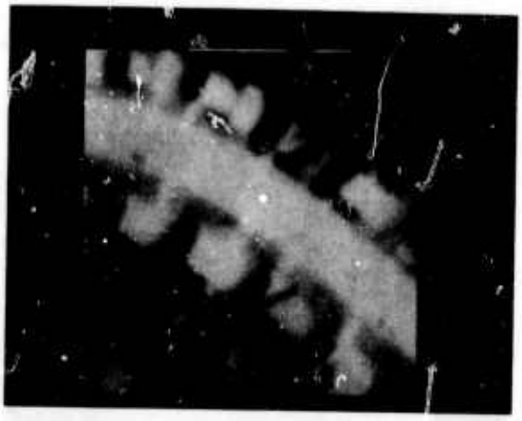


Figure 36. Another suburban scene.

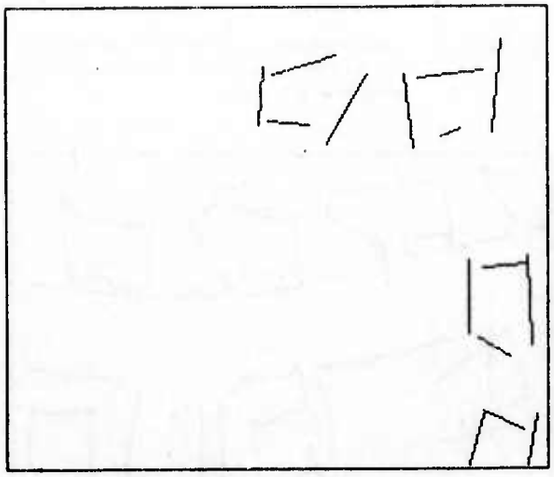


Figure 35. Houses with good confidence

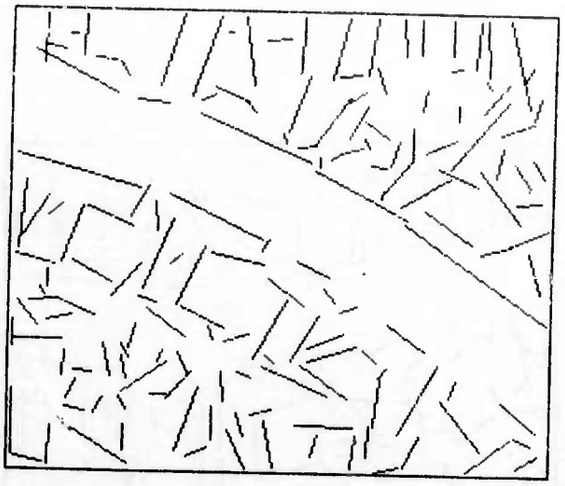


Figure 37. Line segments fitted to the edge components.

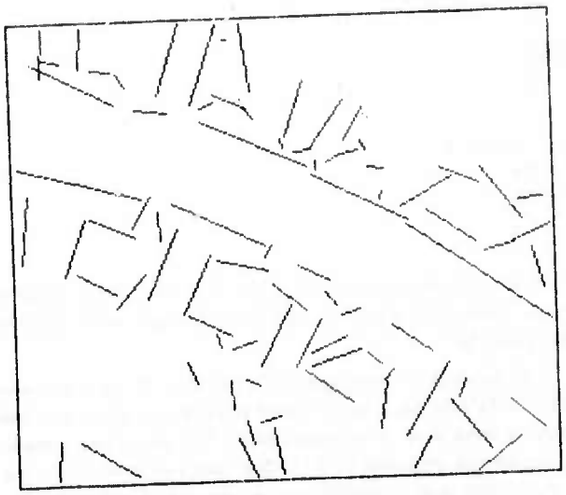


Figure 38. Line segments whose probabilities of being a piece of road or building are nonzero.

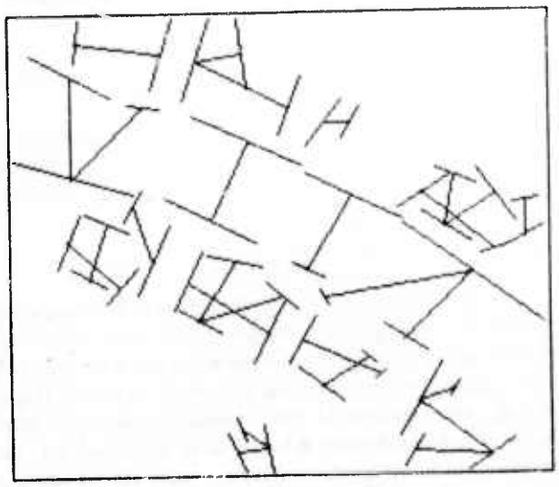


Figure 40. Antiparallel lines.

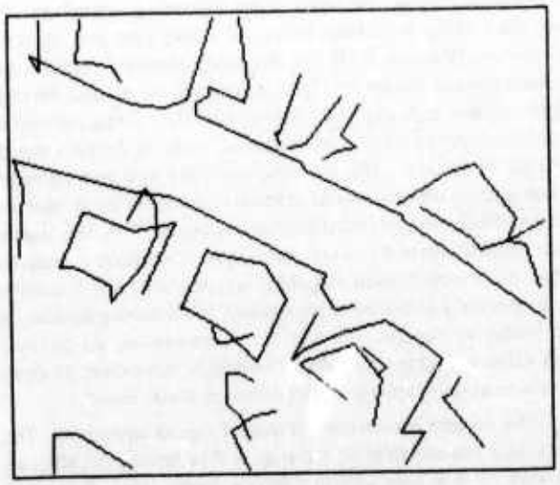


Figure 39. Compatible lines.

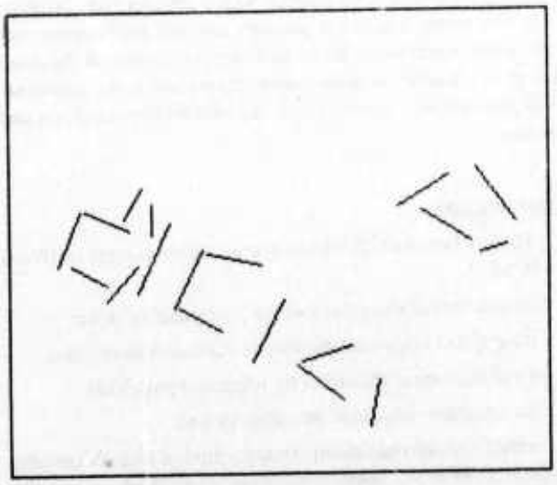


Figure 41. Houses with good confidence.

ATMOSPHERIC MODELLING FOR THE GENERATION OF ALBEDO IMAGES

Robert. W. Sjoberg
Berthold K. P. Horn

MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

ABSTRACT

Accurate classification of terrain is important in the management of natural and other resources. Images obtained from satellite and other high-altitude observations are useful in this task, but only as they provide reflectance information about the surface. In areas of rugged topography, where much of the world's resources are found, the determination of surface reflectance is hampered by interference from the atmosphere and by the presence of cast shadows. The sky serves as a distributed light source which differentially illuminates the surface, depending on local surface orientation. Path radiance, which includes the reflection of sunlight off the atmosphere directly to an observation platform, adds a significant "noise" component to the measured scene radiance. The research reported herein demonstrates that although understanding the effects of the atmosphere is a complex task, the adoption of even simple models can provide substantial improvement over results obtained with no model. In particular, it is shown how the abundance of cast shadows in mountainous regions aids in the determination of path radiance. Several simple sky illumination models are also presented.

1. Introduction

The four factors which interact in any imaging situation are [Woodham 1978a]

- the geometrical arrangement of the objects and the viewer
- the spatial (and spectral) distribution of incident illumination
- the photometry of the surface (its reflectance properties)
- the topography (shape) of the surface viewed.

Armed with the right theory, knowing three of these factors often enables us to obtain the fourth from a given image. Work on extracting shape from shading [Horn 1975; Woodham 1978a; Horn, Woodham, and Silver 1978; Ikeuchi and Horn 1979; Ikeuchi 1979] has been very successful in this endeavor.

One useful variation is the determination of surface reflectance at each point in an image, given the surface shape. Applications to remote sensing immediately come to mind. The essential task facing an image interpreter or an automated classification system in large-scale terrain mapping is to take digitized radiance values for a scene made by high-altitude aircraft or spacecraft, and produce a description of the surface. Much of the earlier interpretation of conventional satellite imagery used radiance values directly as input to a classifier. Some attempts were moderately successful over large agricultural areas in the midwestern United States [Henderson 1975]. Most suffered from large variations in the radiance values which occurred in areas of even moderate relief [Rochon 1978].

If one could correctly account for all effects in the image other than surface reflectance, then it should be possible to create an *albedo image* as input to an automated classifier. The albedo image characterizes intrinsic properties of the surface, independent of local topography, illumination, or satellite position, and should lead to more accurate results. Related work by Horn [Horn 1978] in hillshading has demonstrated the feasibility of creating synthetic images using models of source illumination and surface topography.

In order to correctly model the satellite remote sensing situation, one must consider a fifth factor, one that is not generally significant in terrestrial imaging: the effect of the intervening atmosphere. Its qualitative effects have been known for a long time and are well-documented [Minnaert 1954]. The atmosphere attenuates radiation due to scattering and absorption. To an observer on the ground, the scattered radiation gives rise to the familiar blue sky. To an observer in space, the scattered component contributes "noise" in the form of path radiance or air glow. This scattering also serves as a blurring agent, which redirects reflected ground radiation from outside the imaged target into the observation path [Otterman and Fraser 1979]. The magnitude of these effects depend on wavelength. Absorption is relatively minor in the near-infrared and visible wavelenghs to which satellites in the popular Landsat series are sensitive, but scattering increases at the shorter wavelengths. All four effects—attenuation, sky radiance, path radiance, background radiance—should be represented in an atmospheric model in order to construct accurate albedo maps.

Our research concentrates on areas of rugged topography. This focus stems naturally from the earlier shape from shading and hill shading work. It is in such areas that the atmosphere causes the greatest variations in image data and where albedo maps should provide the most benefit. Mountainous regions are important, since a major use of Landsat imagery is in mapping natural resources, particularly forests and watersheds, which usually occur in areas of significant relief.

2. Development of an Engineering Model

2.1 Role of Radiative Transfer

Any investigation into understanding the nature of atmospheric effects must reckon with the field of radiative transfer through distributed media. The study of radiative transfer is an enormous and

complex domain. Out of early theoretical work on clear and hazy atmospheres [Rayleigh 1871; Schuster 1905; Schwarzschild 1906] grew scattering theory. Astrophysicists studying stellar and planetary atmospheres worked out the first mathematical formulations of the problem as a set of non-linear integrodifferential equations in three dimensions [Chandrasekhar 1960; Sekera 1968]. An analytic solution has been found only for the rather restricted case of isotropic or molecular scattering in a semi-infinite or finite, horizontally homogenous planar atmosphere [Chandrasekhar 1960]. Numerical methods provide inroads to a solution as well as somewhat clearer insight into the physical principles involved. Research in this area, especially applied to satellite remote sensing and communications, has generated a vast literature [Howard, Garing 1971; LaRocca, Turner 1975; Rozenberg 1960], several annual symposia, and a host of multidisciplinary journals.

2.2 Simple Models Permit Engineering Approach

One goal of this research is to construct an *engineering model* of the satellite imaging process. It should be computationally simple, rely as much as possible on the image data itself, very little on *a priori* or *a posteriori* information about particular scenes, and provide adequate accuracy for the application. The full set of radiative transfer equations is far too complicated for our purposes, and unnecessarily cumbersome. On the other hand, numerical methods such as doubling optically thin layers [Hansen 1969] or Monte Carlo techniques [Plass, Kattawar 1968] require knowledge of boundary conditions (including the surface reflectance we are trying to find!), complete specification of the atmospheric state (generally unknown or ill-determined), and take considerable amounts of computation time to yield a result for a single set of conditions. A more productive approach is the use of simple approximate models which are manipulable and have a small number of parameters to be determined by the data.

2.3 Peculiarities of Rugged Terrain

There are certain atmospheric effects in areas of mountainous terrain due to elevation differences and the consequent variation in the amount of air between ground and observation platform. With increasing elevation, one expects

- increasing incident solar irradiance
- increasing reflected (target) radiance
- decreasing path radiance
- decreasing sky irradiance

The first two result because there is less attenuating material at the higher elevations. The last two result because there is less scattering material at higher elevations.

Hill slopes also cause the target irradiance to vary across the image.

- surfaces tilted toward the sun receive more energy than those tilted away
- surfaces with large slope see less of the sky than flat surfaces and are irradiated correspondingly less.

Finally, at the relatively low sun angles typical of Landsat images, there should be significant areas in shadows cast by neighboring hills. Since the sky is the only source of illumination in shadows (both cast shadows and self-shadows), it is important to understand its contribution.

3. Satellite Imaging Equation

3.1 Relation of Reflectance to Satellite Radiance

The image-forming process considered here is shown in Figure 1. The satellite platform is assumed to be at infinity and views the surface normally. Its instantaneous field of view (IFOV) defines a surface element of area A at some altitude z_0 with surface normal making an angle θ_n with respect to the vertical. This element is illuminated directly by the sun, whose extraterrestrial solar irradiance E_0 is attenuated by a factor T_d. Of course, there is no direct illumination if A lies in shadow. The element is also irradiated diffusely by the sky with total sky irradiance E_s. E_s depends on the surface slope, and both E_s and T_d depend in general on altitude.

Assume that the surface is a Lambertian reflector with albedo ρ, $0 \leq \rho \leq 1$ (*albedo* as used here is the bihemispherical reflectance defined in [Nicodemus et al 1977]; the proper bidirectional reflectance distribution function for a Lambertian reflector is $f_r(\theta_i, \phi_i, \theta_r, \phi_r) = \rho/\pi$). Many substances are approximately Lambertian, and the assumption makes calculations reasonable, since the reflected radiance is independent of emittance angle and is proportional to the total irradiance.

The target element A emits a radiance L_t in the zenith direction which is attenuated by the factor T_u, and augmented by the path radiance L_p. The radiance L_m actually measured by the satellite is therefore

$$\begin{aligned} L_m &= L_t T_u + L_p \\ &= \frac{\rho}{\pi}(E_0 T_d S + E_s) T_v + L_p \end{aligned}$$

The factor S accounts for A being in shadow and for the reduced solar irradiance due to foreshortening.

$$S = \begin{cases} 0. & \text{if pixel is in cast shadow;} \\ \cos\theta_{i0}. & \text{if } \cos\theta_{i0} > 0; \\ 0. & \text{otherwise.} \end{cases}$$

θ_{i0} is the angle between the surface normal in polar direction (θ_n, ϕ_n) and the sun's direction (θ_0, ϕ_0).

$$\cos\theta_{i0} = \cos\theta_n \cos\theta_0 + \sin\theta_n \sin\theta_0 \cos(\phi_n - \phi_0)$$

(The reference line for azimuthal angles ϕ is arbitrary, but lies in the plane of the earth's surface.) The albedo of the surface at the imaged point can be written

$$\rho = \frac{\pi(L_m - L_p)}{(E_0 T_d S + E_s) T_u}$$

We assume that L_m (converted from its digitized value to the appropriate radiance in mw cm^{-2}sr^{-1}), E_0, and S (computed from the shadow and surface slope models) are available. The remaining four parameters—T_d, T_u, L_p, and E_s—depend on the actual atmospheric model adopted.

3.2 General Assumptions

The discussion which follows assumes geometrical ray optics, a point source, distant source and observer, and monochromatic light or the equivalent band-averaged quantity. It ignores polarization, atmospheric absorption, and mutual illumination of surface elements by neighboring ones as in valleys.

3.3 Use of Topographic Models

The region chosen for the study is a 17.5 km by 24.0 km part of southwestern Switzerland for which a digital elevation model (DEM) and Landsat images had been obtained. The DEM is an array of 175 by 240 elevation values (vertical quantization is 10 meters) on a 100-meter grid. Elevations range from 410 m in the Rhone River valley to 3210 m on the summit of des Diablerets. Satellite data is a portion of Landsat 1 scene 1078-09555 acquired about 9:55 GMT October 9, 1972. The sun at that time had an elevation of 34.2 degrees, and an azimuth of 154.8 degrees [Horn 1978; Horn,Woodham 1979b]. Radiometric corrections were necessary to remove regular striping effects [Horn, Woodham 1978a]. The scene was then registered to the DEM using the method described in [Horn, Bachman 1978]. Figure 2 shows multispectral scanner band 4 (MSS 4, 0.5–0.6μm, green-yellow) for the scene.

A digital slope model was constructed from the DEM by using modified first-differences [Horn 1979]. Cast shadows were generated by application of a hidden-surface-removal algorithm using the sun as the viewer [Woodham 1976, 1978b]. Figure 3 shows the terrain as if it were a perfect Lambertian reflector (unit albedo) illuminated by a point sun in the same position as during the Landsat overflight. Cast shadow information is superimposed on the synthetic image, and is also shown in Figure 4. A black point indicates the pixel is in shadow.

An albedo map constructed with no path radiance or sky illumination modeling is shown in Figure 5. Because of the path radiance, even surface points in shadow will have non-zero data values. Consequently, since their irradiance is zero, they must have infinite reflectance (or be surface points in shadow will have non-zero data values. Consequently, since their irradiance is zero, they must have infinite reflectance (or be light sources). It is not possible using such a primitive model to distinguish even snow and cloud from shadowed regions.

4. Modeling Path Radiance

4.1 Constant Path Radiance Model

A very simple model of path radiance is to assume a constant value across the image. This may be obtained by using the lowest data value in the scene [Fraser and Curran 1976], and is appropriate for flat areas over which the atmosphere does not change radically. Figure 6 illustrates this model for our mountainous region. The effect is simply to shift all radiance values down the same amount (about 0.22 mw cm^{-2}sr^{-1}here), and remove the hazy appearance of the image.

4.2 Elevation-Dependent Path Radiance

A more realistic model of path radiance can be derived if we assume at least first-order dependence on the atmosphere. The real atmosphere has two distinct components. Atmospheric gas molecules scatter light almost isotropically (within a factor of two) and are distributed throughout the entire air column with relatively constant mixing ratios. Molecular scattering accounts for the blue sky and was first explained by Lord Rayleigh [Rayleigh 1871]. Atmospheric aerosols form the other component, and include all particulate matter (smoke, dusts, sea spray, rain, etc.). Their scattering behavior was first investigated by Mie [Mie 1908], and is highly anisotropic: about 95% of the scattered radiation is emitted in a narrow cone in the forward direction, with most of the remaining 5% emitted in a narrow backward cone. The aerosols vary considerably depending on locale, even across the area of a single satellite image. They are distributed close to the surface and are responsible for most of the scattering in the lower 1 or 2 km of air. The reader is referred to the literature (such as [McCartney 1976] and the references cited therein) for a detailed description of molecular (also called Rayleigh) and aerosol scattering in the earth's atmosphere.

For computational simplicity, we assume an atmosphere which is planar, horizontally uniform and infinite, and composed of two kinds of non-absorbing scatterers, one responsible for near-surface scattering (called the aerosol component), the other for higher altitudes (called the Rayleigh component). Each form is assumed to scatter isotropically and to be distributed exponentially with altitude throughout the air column. Each component is characterized by an extinction coefficient $\beta(z)$ which gives the fraction of radiation scattered out of the incident beam per unit length of travel through the medium. If we further assume that only single scattering is significant, then the two components can be treated independently and added together for the total effect. Multiple scattering is known to occur and can be significant [Pollack 1967; Pollack et al 1977], but the assumption is important to ease computation. Neglect of multiple scattering also eliminates consideration of the influence of background radiance (light originating from outside the field of view which is scattered into the sensor window by the air). In some circumstances, this one effect dominates the path radiance [Turner 1974].

Because absorption is ignored, the extinction coefficients give both the amount of radiation which is extinguished from an incident beam and the fraction expected to go into skylight and path radiance. The coefficients are exponentials typical of an isothermal atmosphere.

$$\beta_R(z) = \beta_R(0)e^{-z/H_R} \qquad \text{and} \qquad \beta_A(z) = \beta_A(0)e^{-z/H_A}$$

The reference altitude is arbitary, but we chose $z = 0$ at sea level. H_R and H_A are the *scale heights* for the two respective scatterers. The path radiance expected over a surface element at altitude z_0 for each component then has the form

$$L_p(z_0) = \frac{1}{4\pi} E_0 \frac{1}{q}(1 - e^{-q\tau(z_0)})$$

where E_0 is the extraterrestrial solar irradiance, $q = 1 + \sec\theta_0$, and $\tau(z_0)$ is the *optical depth* to altitude z_0, defined by

$$\begin{aligned}\tau(z_0) &\equiv \int_{z_0}^{\infty} \beta(z)\,dz \\ &= \int_{z_0}^{\infty} \beta(0)e^{-z/H}\,dz \\ &= \beta(0)He^{-z/H} = \tau(0)e^{-z/H}\end{aligned}$$

Vertical transmission from any altitude up to the observation platform and slant path transmission from the sun down can be written in terms of the optical depth as well:

$$T_u(z) = e^{-\tau(z)} \qquad \text{and} \qquad T_d(z) = e^{-\tau(z)\sec\theta_0}$$

The total path radiance is the sum of the two component radiances

$$L_p(z_0) = L_p^R(z_0) + L_p^A(z_0)$$
$$= \frac{1}{4\pi} E_0 \frac{1}{q} [2 - \exp(-q\tau_R(z_0)) - \exp(-q\tau_A(z_0))]$$

4.3 Finding Path Radiance Parameters

For our two-component model, there are four parameters to determine from the image data. One method is to use clear water lakes as standard, low-albedo reflectors [Ahern et al 1977]. If they were ideal absorbers, the measured radiance would indeed be all path radiance. Unfortunately, not all images will have sufficiently many clear water lakes, nor are the lakes likely to be distributed conveniently through the altitude range.

Areas in cast shadows, on the other hand, are common in mountainous regions, and provide an alternative way to determine path radiance. Shadowed pixels are generally scattered throughout the image at all but the highest elevations. They are illuminated only by the sky, but if their albedo is small, one may take the recorded satellite data as a fair value of the path radiance for the corresponding elevation.

Figure 7 shows a scattergram of the radiance values for all shadowed pixels in the MSS 4 image. The straight line at the bottom is from the constant path radiance model mentioned earlier. The curve is considered a good fit to the data for the $L_p(z_0)$ equation given above. The parameter values are $\tau_R(0) = 0.0984$, $\tau_A(0) = 0.1834$, $H_R = 9500$ m, and $H_A = 1800$ m. The curve is not very sensitive to the precise choice of parameters. The optical depths at sea-level were taken from tabulated values for molecular and aerosol components [Valley 1965] and the scale heights adjusted for an eyeball fit (a slight gap was left since the surface does not really have zero albedo). There is no point to modelling the upward turn of the minimum radiances at higher elevations, since these pixels are mostly covered with snow. Their high albedo results in large values of reflected sky radiation even though the sky is not as bright. There is also considerable radiance due to mutual illumination and multiple scattering from the surrounding bright background.

No image is included here to illustrate this improved path radiance model. The differences between it and the constant model (Figure 6) are too subtle to survive the printing process. The major difference is the appearance of slightly darker valleys. This is to be expected, since subtracting a single minimum value from every point in the image fails to compensate adequately for the larger path radiance at lower altitudes.

The exponential model of path radiance is used throughout the remainder of this paper, although it is unlikely that large discrepancies would result from using a constant value instead.

5. Models of Sky Radiance

5.1 Effects of Sky on Scenes with Mountain Slopes

The slopes found in rugged terrain certainly affect image formation. A tilted surface is exposed differentially to the sky, and receives more or less radiation as the magnitude of its slope changes. Consider an element of surface at altitude z_0 with surface normal pointing in direction (θ_n, ϕ_n). Let the distribution of sky radiance for an observer at altitude z_0 be given by $L_s(z_0, \theta, \phi)$. Then the total irradiance due to sky illumination is [Horn and Sjoberg 1979]

$$E_s(z_0, \theta_n, \phi_n) = \int_{-\pi}^{\pi} \int_0^{\theta_H(\phi)} L_s(z_0, \theta, \phi) \cos\theta_i \sin\theta \, d\theta \, d\phi$$

The geometry is illustrated in Figure 8. The double integral is over the entire hemisphere of sky to which the surface element is exposed. It is implicit in the equation that only downwelling radiance from the sky is included, that is, L_s for $\theta > \pi/2$ is zero. The factor $\cos\theta_i$ accounts for the foreshortening of the surface as the integration direction deviates from (θ_n, ϕ_n) and is given by

$$\cos\theta_i = \cos\theta_n \cos\theta + \sin\theta_n \sin\theta \cos(\phi_n - \phi)$$

The upper limit on polar angle θ is a non-trivial function of the azimuthal angle ϕ, and reflects the situation illustrated in Figure 9 where part of the sky behind the surface element is cut off.

$$\cot\theta_H(\phi) = \begin{cases} \pi/2, & \text{if } |\phi| < \pi/2 \\ -\tan\theta_n \cos(\phi_n - \phi), & \text{if } |\phi| > \pi/2 \end{cases}$$

Except for relatively simple expressions of L_s, the sky irradiance integral is exceedingly difficult to solve. Fortunately, the assumption of isotropic scattering implies that all sky radiance functions we consider are independent of azimuthal angle ϕ. It also means that irradiance from the sky on any surface element is independent of the azimuthal angle ϕ_n.

5.2 Uniform Hemispherical Sky

If the sky radiance function L_s is a constant, then the sky irradiance integral is easily evaluated [Horn and Sjoberg 1979; Brooks 1978; Moon 1942]

$$E_s(z_0, \theta_n) = \frac{\pi}{2} L_s (1 + \cos\theta_n)$$

This function was used to generate the albedo image shown in Figure 10. The brightest areas correspond to albedos of 0.73 or larger, dark areas to 0.0. The constant L_s was chosen to be 0.83 mw cm^{-2}sr^{-1}, a value calculated so that a flat surface would have an irradiance comparable to measured values [Rogers 1973]. The improvement over the naively made albedo image is striking. For one thing, shadows now contain useful information that was not appparent in the original Landsat data. Snow fields on the glacier in the northeast (upper left) corner which are shadowed are visible as high-albedo regions. Much of the terrain now appears somewhat flattened because the shading variations which give clues to shape are diminished or eliminated. Of course, there are many isolated high-albedo points which result from noise in the data, especially on the ridge lines, where slope information is very coarse or outright wrong.

5.3 Elevation-Dependent Hemispherical Sky

One improvement to the uniform hemispherical sky model made at only a slight cost in computation is to make the sky radiance L_s a function of elevation. The sky is brighter at lower altitudes, and sky irradiance should be greater. A simple model for elevation-dependent sky radiance is $L_s(z_0, \theta) = CL_z(z_0)$, where $L_z(z_0)$ is the zenith sky radiance measured by an observer at elevation z_0. Except at extreme sun angles, the zenith sky radiance differs from the path radiance by only a few percent. C is a constant which may be varied for the best effect.

Figure 11 is an albedo map for $C = 3.74$, a value selected so that the total irradiance of a flat element at an average altitude would match experimental values. There are no readily apparent differences between the altered model and the constant one. Close examination reveals that points at low altitudes have slightly lower albedos with the elevation-dependent model, due to the increased sky irradiance. Similarly, points at high altitudes have slightly higher albedos because of the reduced sky irradiance.

5.4 Sky Radiance for an Infinite Planar Sky

The two previous sky models are easy to compute, but do not reflect a very realistic atmosphere. To an observer on the ground, the real sky becomes considerably brighter toward the horizon. This suggests one regard the atmosphere as the plane-parallel, horizontally homogeneous and infinite model described in section 4.2 above. Suppose that the sky radiance in direction (θ, ϕ) were proportional to the number of scatterers in the slant path. Since this varies as the secant of the zenith angle, $Ls(z_0, \theta) = L_z(z_0) \sec\theta$. The constant of proportionality here is once again the zenith radiance. For a flat surface, this form of L_s integrates out to $2\pi L_z(z_0)$. For tilted surfaces, the integral blows up as θ approaches $\pi/2$. It is better to adopt a bounded sky radiance.

$$L_s(z_0, \theta) = \begin{cases} L_z(z_0) \sec\theta, & \text{if } \theta < \theta_m \\ L_m \equiv L_z(z_0) \sec\theta_m, & \text{if } \theta \geq \theta_m \end{cases}$$

With a suitable approximation in the integral, the sky irradiance may be worked out.

$$E_s(z_0, \theta_n) = 2L_z(z_0)\left(C_1 \cos\theta_n + C_2 \sin\theta_n + (\frac{\pi}{2} - \theta_n)\cos\theta_n - \frac{1}{3}\cos^2\theta_n \sin\theta_n \right)$$

where the constants C_1 and C_2 depend on θ_m:

$$C_1 = \frac{\pi}{2} - \frac{\pi}{4}\cos\theta_m$$

$$C_2 = \frac{1}{2}\left(\frac{\frac{\pi}{2} - \theta_m}{\cos\theta_m} - \sin\theta_m - \ln\frac{1 - \sin\theta_m}{1 + \sin\theta_m} \right)$$

Figure 12 shows the above sky irradiance relative to the zenith radiance as a function of θ_n for $\theta_m = 85$ degrees. For a given θ_n, the variation with altitude is identical to the dependence of the zenith radiance (that is to say the path radiance, which is almost the same). An albedo map constructed using this form for sky irradiance is shown in Figure 13. Once again, there are no striking differences between it and the previous two models. There seems to be slightly better definition of some of the lower areas, and more snow cover is picked up with the secant model.

5.5 More Sophisticated Models

The form for E_s developed in the previous section indicates how solving the sky irradiance equation can be difficult and tedious. Progressively more sophisticated models prove disproportionately more awkward, even with approximate methods. We are continuing research into more realistic yet (we hope) computationally tractable formulations.

6. Evaluation

It is clear that some sort of atmospheric correction must be applied before albedo maps can be reliably generated for mountainous terrain. Cast shadows can be used to find parameters for path radiance. The path radiance so determined may be used in the expressions for sky irradiance. Even simple models of path and sky radiance reveal more than is apparent in the raw satellite image. If the results obtained so far in this research are a good indication, the important conclusion may not be so much in the choice of model, but that *some* model of atmospheric effects be incorporated into the interpretation phase.

7. References

F. J. Ahern, D. G. Goodenough, S. C. Jain, and V. R. Rao (1977). Use of clear lakes as standard reflectors for atmospheric measurements. *Proc. 11th Intl. Symp. Remote Sensing of the Environment*, Ann Arbor, 25–29 April, pp. 731–755.

M. J. Brooks (1978). Investigating the effects of planar light sources. CSM 22, Department of Computer Science, Essex University, Colchester, England.

S. Chandrasekhar (1960). *Radiative Transfer*. New York: Dover Publications.

R. S. Fraser and R. J. Curran (1976). Effects of the atmosphere on remote sensing. Chapter 2 of *Remote Sensing of Environment*, J. Lintz, Jr. and D.S. Simonett, eds. Reading: Addison-Wesley.

R. G. Henderson (1975). Signature extension using the MASC algorithm. *Proc. Second Symp. on Machine Processing of Remotely Sensed Data*, W. Lafayette, IN, 3–5 June.

M. D. Fleming and R. M. Hoffer (1979). Machine processing of LANDSAT MSS data and DMA topographic data for forest cover type mapping. *Proc. Fifth Symp. on Machine Processing of Remotely Sensed Data*, West Lafayette, IN, 27–29 June, pp. 377–390.

R. M. Hoffer and staff (1975). *An Interdisciplinary Analysis of Colorado Rocky Mountain Environments Using ADP Techniques*. Alternate title: *Natural Resource Mapping in Mountainous Terrain by Computer Analysis of ERTS-1 Satellite Data*. Laboratory for Applications of Remote Sensing Information Note 061575, Purdue University, June.

B. K. P. Horn (1975). Determining shape from shading. Chapter 4 of *The Psychology of Computer Vision*, P. Winston, ed. New York: McGraw-Hill.

B. K. P. Horn (1978). The position of the sun. Working Paper 162, Artificial Intelligence Laboratory, M.I.T., March.

B. K. P. Horn (1979). Hill-shading and the reflectance map. *Proceedings: Image Understanding Workshop*, Palo Alto, CA, 24–25 April, pp. 79–120.

B. K. P. Horn and B. L. Bachman (1978). Using synthetic images to register real images with surface models. *CACM*, **21**:914–924, November.

B. K. P. Horn and R. W. Sjoberg (1979). Calculating the reflectance map. *Appl. Op.*, **18**:1770–1779, 1 June.

B. K. P. Horn and R. J. Woodham (1979a). Destriping LANDSAT MSS images by histogram modification. *Computer Graphics and Image Processing*, **10**:69–83.

B. K. P. Horn and R. J. Woodham (1979b). LANDSAT MSS coordinate transformations. *Proc. Fifth Symp. on Machine Processing of Remotely Sensed Data*, W. Lafayette, IN, 27–29 June, pp. 59–68.

B. K. P. Horn, R. J. Woodham, and W. M. Silver (1978). Determining shape and reflectance using multiple images. A.I. Memo 490, Artificial Intelligence Laboratory, M.I.T., August.

J. N. Howard and J. S. Garing (1971). Atmospheric optics and radiative transfer. *Trans. Am. Geophys. Union*, 52:371–389, June.

K. Ikeuchi (1979). Numerical shape from shading and occluding contours in a single view. A.I. Memo 566, Artificial Intelligence Laboratory, M.I.T., November.

K. Ikeuchi and B. K. P. Horn (1979). An application of the photometric stereo method. A.I. Memo 539, Artificial Intelligence Laboratory, M.I.T., August.

A. J. LaRocca and R. E. Turner (1975). *Atmospheric Transmittance and Radiance: Methods of Calculation*. IRIA State-of-the-Art Report, Environmental Research Institute of Michigan, report no. ERIM 107600-10-T, June.

E. J. McCartney (1976). *Optics of the Atmosphere*. New York: John Wiley and Sons.

R. A. McClatchey, R. W. Fenn, J. E. A. Selby, F. E. Volz, J. S. Garing (1978). optical properties of the atmosphere. Section 14 of *Handbook of Optics*, W. G. Driscoll, ed. New York: McGraw-Hill.

G. Mie (1908). A contribution to the optics of turbid media, especially colloidal metallic suspension. *Ann. Phys.*, 25:377–445.

M. Minnaert (1954). *The Nature of Light and Color in the Open Air*. New York: Dover.

P. Moon and D. E. Spencer (1942). Illumination from a non-uniform sky. *Illuminating Engineering*. XXXVII:707–726, December.

F. E. Nicodemus, J. C. Richmond, J. S. Hsia, I. W. Ginsberg, T. Limperis (1977). *Geometrical Considerations and Nomenclature for Reflectance*. National Bureau of Standards, NBS Monograph 160, U.S. Department of Commerce.

G. N. Plass and G. W. Kattawar (1968). Calculations of reflected and transmitted radiance for earth's atmosphere. *Appl. Op.*, 7:1129–1135, June.

J. B. Pollack, D. Colburn, R. Kahn, J. Hunter, W. Van Camp, C. E. Carlston, M. R. Wolf (1977). Properties of aerosols in the Martian atmosphere as inferred from Viking lander imaging data. *J. Geophys. Res.*, 82:4479–4496, September 30.

J. B. Pollack (1967). Rayleigh scattering in an optically thin atmosphere and its application to Martian topography. *Icarus*, 7:42–46.

Lord Rayleigh (1871). On the scattering of light by small particles. *Phil. Mag.*, 41:447–454.

G. Rochon, H. Audirac, F. J. Ahern, and J. Beaubien (1978). Analysis of a transformation model of satellite radiances into reflectances. *Proc. 12th Intl Symp. Remote Sensing of the Environment*, Manila, Philippines, 20–26 April.

R. H. Rogers (1973). *Investigation of Techniques for Correcting ERTS Data for Solar and Atmospheric Effects*. National Aeronautics and Space Administration, Report NASA-CR-132860, August.

R. H. Rogers, K. Peacock, and J. J. Shah (1973). A technique for correcting ERTS data for solar and atmospheric effects. *Third Symp. on Significant Results Obtained from ERTS-1*, Washington, DC, 10–14 December, NASA Report SP-351, pp. 1787-1804

G.V. Rozenberg (1960). Light scattering in the earth's atmosphere. *Soviet Physics Uspekhi*, 3:346–371, Nov.-Dec.

A. Schuster (1905). Radiation through a foggy atmosphere. *Astrophys. J.*, 21:1–22.

K. Schwarzschild (1906). *Gottinger Nachrichten*, 41.

Z. Sekera (1968). Radiative transfer and the scattering problem. *J. Quant. Spectrosc. Rad. Trans.*, 8:17–24.

R.E. Turner (1974). *Radiative Transfer in Real Atmospheres*. Environmental Research Institute of Michigan Report ERIM 19100-24-T, July.

S. L. Valley (1965). *Handbook of Geophysics and Space Environments*. New York: McGraw-Hill.

R. J. Woodham (1976). Two simple algorithms for displaying orthographic projections of surfaces. Artificial Intelligence Laboratory Working Paper No. 126, M.I.T., August (unpublished).

R. J. Woodham (1978a). Reflectance map techniques for analyzing surface defects in metal castings. Artificial Intelligence Laboratory Technical Report TR-457, M.I.T., June.

R. J. Woodham (1978b). Looking in the shadows. Artificial Intelligence Laboratory Working Paper No. 169, M.I.T., May (unpublished).

Figure 1. Imaging geometry for a satellite over rugged terrain. The normal to the surface element makes an angle θ_n with the vertical. Illumination is from distant sun at angle θ_0 and from sky overhead.

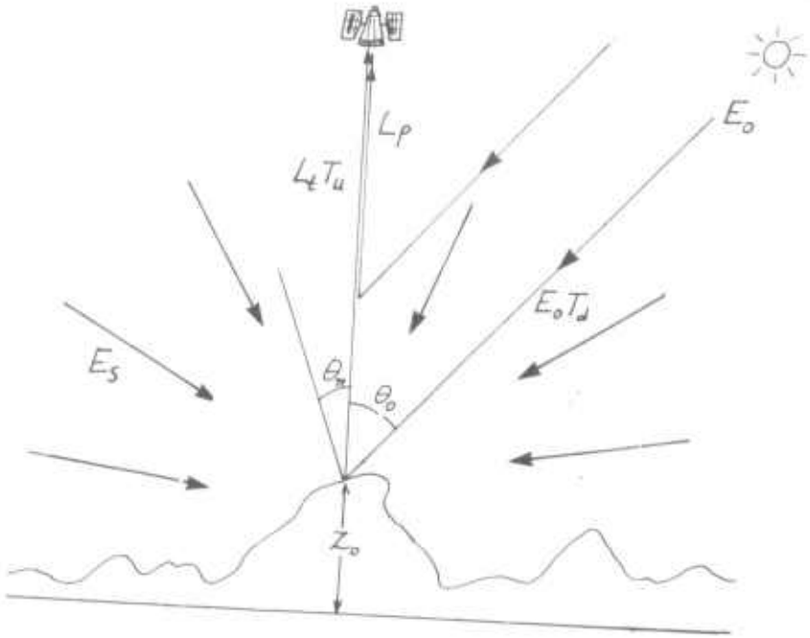


Figure 2. A portion of band 4 of the Landsat 1 multispectral scanner from its flight over the part of Switzerland described in the text. Sun elevation is 34.2 degrees, azimuth 154.8 (counterclockwise from North).

Figure 3. A synthetic image of the Switzerland area. The surface is a perfect Lambertian reflector, illuminated by a distant point sun from the same direction as during the Landsat overflight. Cast shadows are included.

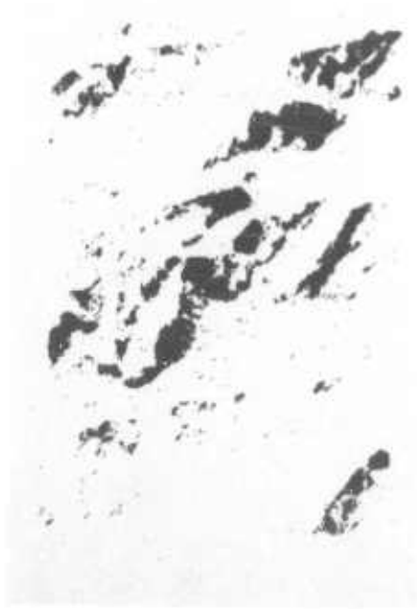
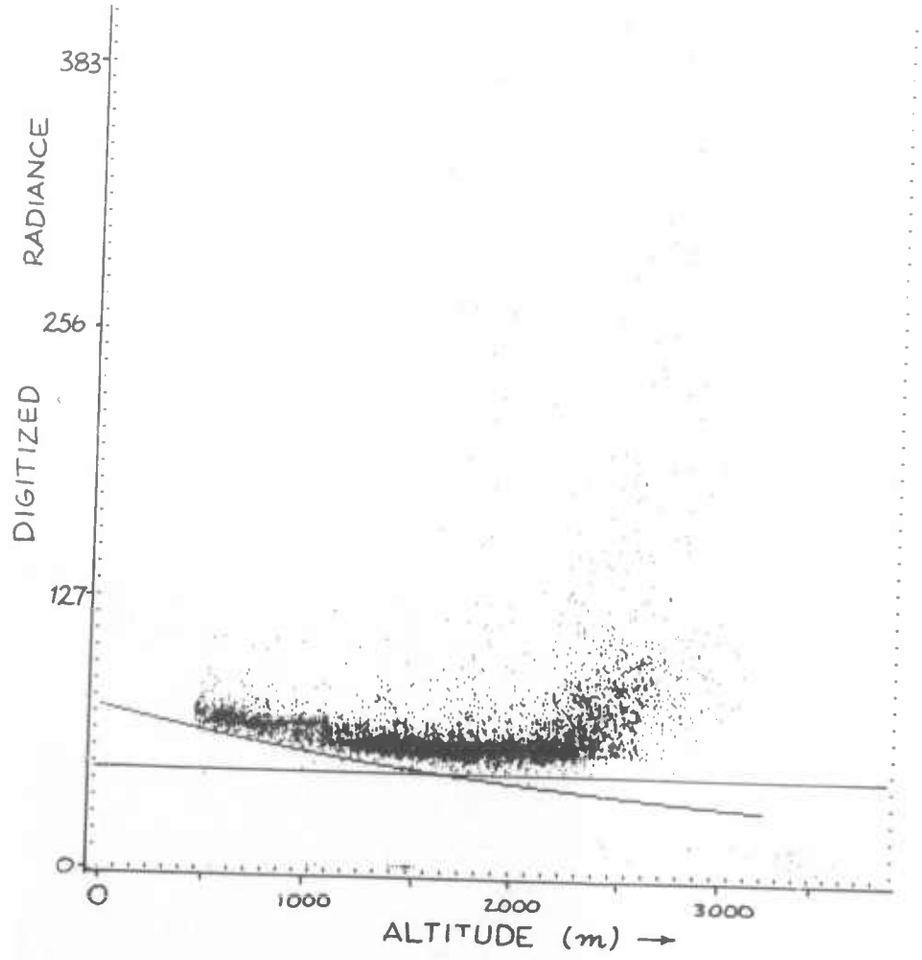


Figure 4. Locations of all pixels which are in shadow for the sun position given in the text.

Figure 5. Primitive albedo map generated as the ratio of actual satellite radiances to those predicted by the synthetic image. Shadowed areas show up as infinitely reflective.

Figure 6. The result of subtracting a constant value from the original radiance data. The removal of the haze layer causes an increase in contrast.

Figure 7. Scattergram of radiance values from the Landsat data for those pixels determined to lie in shadow. Vertical axis is digitized radiance (a value of 511 corresponds to 2.48 mw $cm^{-2}sr^{-1}$). The horizontal axis is elevation, in 10 m intervals. The straight line represents a constant path radiance level, and the curve is the altitude-dependent model described in the text.



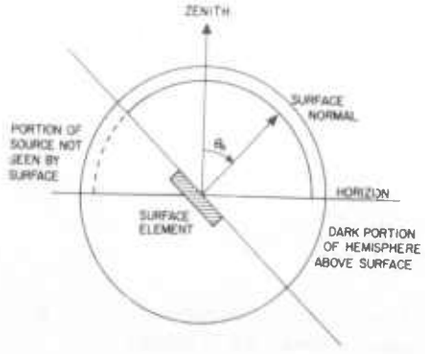


Figure 9. A tilted surface is exposed only to part of the hemisphere overhead.

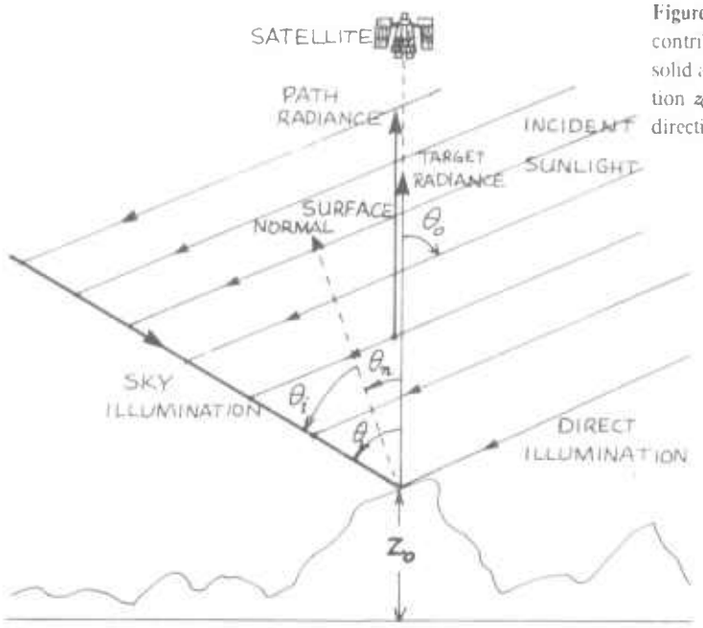


Figure 8. The geometry of sky irradiance. The sky in direction (θ, ϕ) contributes a portion $L_s(z_0, \theta, \phi)\, d\omega = L_s(z_0, \theta, \phi) \sin\theta\, d\theta\, d\phi$ through solid angle $d\omega$ to the total sky irradiance of the surface element at elevation z_0. *theta*, is the angle between the surface normal and the sky direction

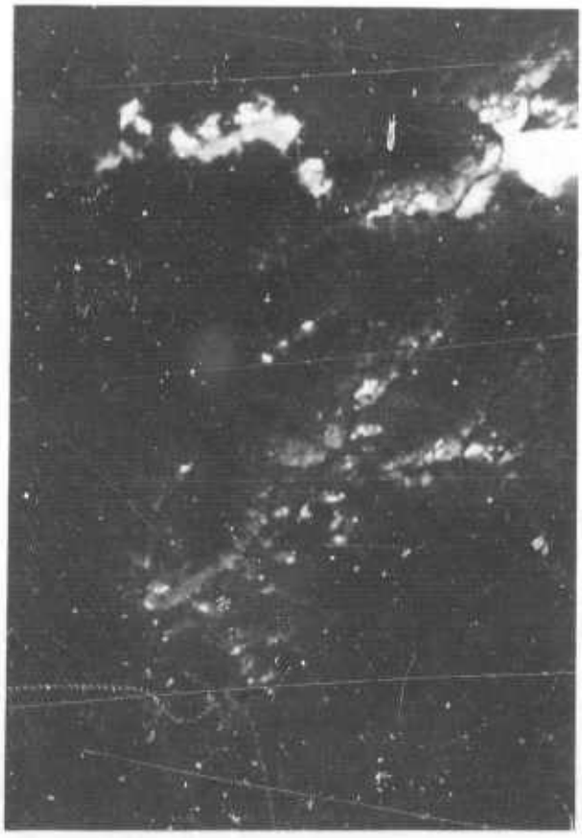


Figure 10. Albedo image for a uniform hemispherical sky. The brightest pixels correspond to albedos of 0.73 or larger.

Figure 11. Albedo image for an elevation-dependent but otherwise uniform hemispherical sky. Bright areas have albedo 0.73 or larger.

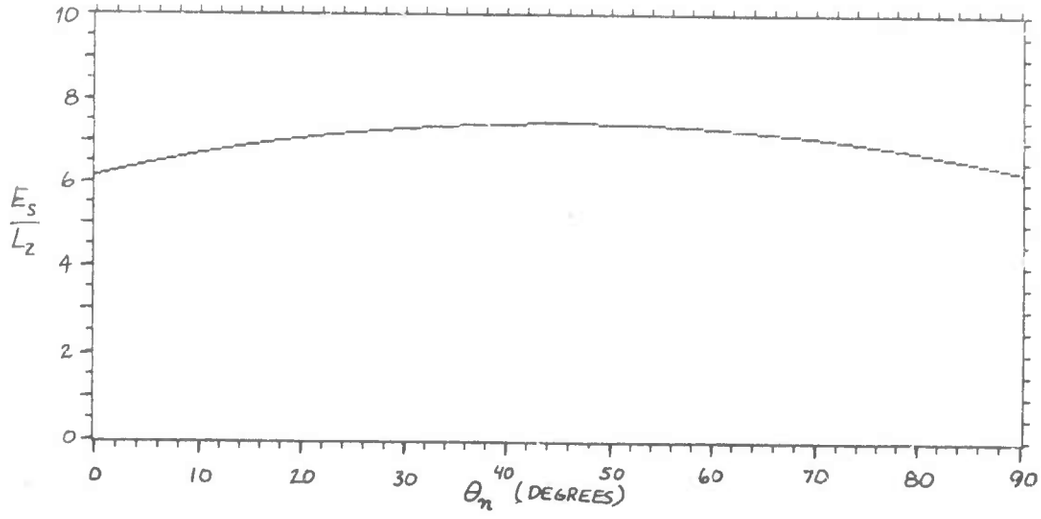


Figure 12. Form of sky irradiance $E_s(z_0, \theta_n)$ for the infinite planar sky under assumption that $L_s(z_0, \theta)$ varies as $\sec \theta$. Horizontal axis is θ_n, the tilt of the surface with respect to the zenith. Vertical axis is radiance relative to zenith radiance $L_z(z_0)$.

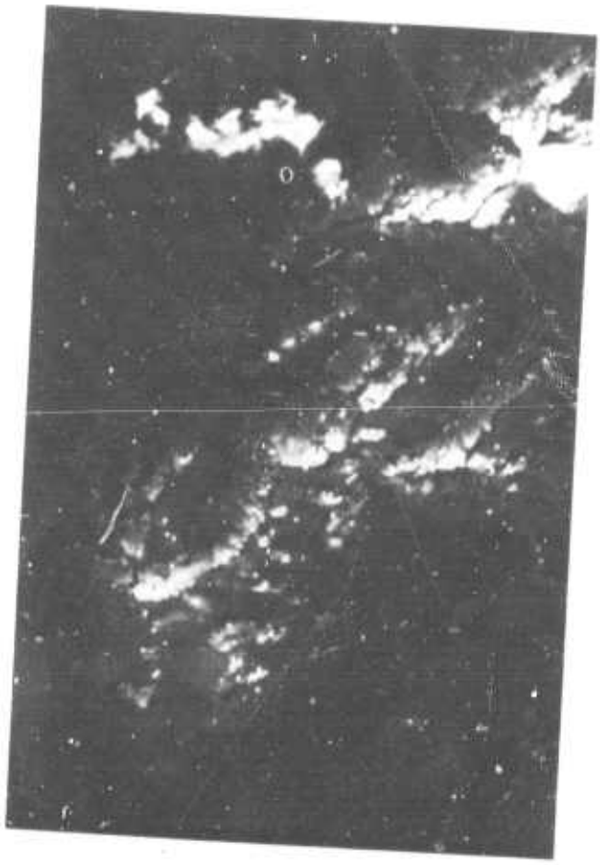


Figure 13. Albedo map using the sky irradiance shown in Figure 12. Brightest areas have albedo 0.73 or larger.

RANDOM SAMPLE CONSENSUS: A PARADIGM FOR MODEL FITTING WITH APPLICATIONS TO IMAGE ANALYSIS AND AUTOMATED CARTOGRAPHY

M. A. Fischler and R. C. Bolles

Artificial Intelligence Center
SRI International
Menlo Park, California 94025

ABSTRACT

In this paper we introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and thus is ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. We derive new results on the minimum number of landmarks needed to obtain a solution, and present algorithms for computing these minimum-landmark solutions in closed form. These results provide the basis for an automatic system that can solve the LDP under difficult viewing and analysis conditions. Implementation details and computational examples are also presented.

INTRODUCTION

In this paper we introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data: and we illustrate its use in scene analysis and automated cartography. The application discussed, the location determination problem (LDP), is treated at a level beyond that of a mere example of the use of the RANSAC paradigm; we present new basic findings concerning the conditions under which the LDP can be solved and describe a comprehensive approach to the solution of this problem that we anticipate will have near-term practical applications.

To a large extent, scene analysis (and in fact, science in general) is concerned with the interpretation of sensed data in terms of a set of predefined models. Conceptually, interpretation involves two distinct activities: first, there is the problem of finding the best match between the data and one of the available models (the classification problem); second, there is the problem of computing the best values for the free parameters of the selected model (the parameter estimation problem). In practice, these two problems are not independent--a solution to the parameter estimation problem is often required to solve the classification problem.

Classical techniques for parameter estimation, such as "least squares," optimize (according to a specified objective function) the fit of a functional description (model) to ALL of the presented data. These techniques have no internal mechanisms for detecting and rejecting gross errors. They are averaging techniques that rely on the assumption (the "smoothing assumption") that the maximum expected deviation of any datum from the assumed model is a direct function of the size of the data set, and thus regardless of the size of the data set, there will always be enough "good" values to "smooth out" any gross deviations.

In many practical parameter estimation problems the smoothing assumption does not hold: that is, the data contains uncompensated gross errors. To deal with this situation, a number of heuristics have been proposed. The technique usually employed is some variation of the idea of first using all the data to derive the model parameters: next, locate the datum that is farthest from agreement with the instantiated model, assume that it is a gross error, delete it, and iterate this process until either the maximum deviation is less then some preset threshold, or until there is no longer sufficient data to proceed.

It can easily be shown that a single gross error ("poisoned point"), mixed in with a set of good data, can cause the above heuristic to fail (for example, see Figure 1). It is our contention that averaging is not an appropriate technique to apply to an "unverified" data set.

In the following section we introduce the RANSAC paradigm, which is capable of smoothing data that contains a significant percentage of gross errors. This paradigm is particularly applicable to scene analysis because local feature detectors, which often make mistakes, are the source of the data provided to the interpretation algorithms. Local feature detectors make two types of errors--classification errors and measurement errors. Classification errors occur when a feature detector incorrectly identifies a portion of an image as an occurrence of a feature. Measurement errors occur when the feature detector correctly identifies the feature, but slightly miscalculates one of its parameters (e.g., its image location). Measurement errors generally follow a normal distribution, and therefore the smoothing assumption applies to them. Classification errors, however, are gross errors because they have a significantly larger effect than measurement errors and they do not average out.

In the final sections of this paper we discuss the application of RANSAC to the location determination problem:

> Given a set of "landmarks" ("control points"), whose locations are known in some coordinate frame, determine the location (relative to the coordinate frame of the landmarks) of that point in space from which an image of the landmarks was obtained.

We first derive some new results on the minimum number of landmarks needed to obtain a solution, and then present algorithms for computing these minimum-landmark solutions in closed form. (Conventional techniques are iterative and require a good initial guess to assure convergence.) These results form the basis for an automatic system that can solve the LDP under severe viewing and analysis conditions. In particular, the system performs properly even if a significant number of the landmarks are incorrectly located due to low visibility, terrain changes, or image analysis errors. Implementation details and experimental results are presented to complete our description of the LDP application.

RANDOM SAMPLE CONSENSUS

The philosophy of RANSAC is opposite to that of conventional smoothing techniques--rather than using as much of the data as possible to obtain an initial solution and then attempting to eliminate the invalid data points, RANSAC uses as small an initial data set as feasible and enlarges this set with consistent data when possible. For example, given the task of fitting an arc of a circle to a set of two-dimensional points, the RANSAC approach would be to select a set of three points (since three points are required to determine a circle), compute the center and radius of the implied circle, and count the number of points that are close enough to that circle to suggest their compatibility with it (i.e., their deviations are small enough to be measurement errors). If there are enough compatible points, RANSAC would employ a smoothing technique, such as least squares, to compute an improved estimate for the parameters of the circle now that a set of mutually consistent points has been identified.

The RANSAC paradigm is more formally stated as follows:

> Given a model that requires a minimum of n data points to instantiate its free parameters, and a set of data points P such that the number of points in P is greater than n (#(P) > n), randomly select a subset S1 of n data points from P and instantiate the model. Use the instantiated model M1 to determine the subset S1* of points in P that are within some error tolerance of M1. The set S1* is called the consensus set of S1.
>
> If #(S1*) is greater than some threshold t, which is a function of the estimate of the number of gross errors in P, use S1* to compute (possibly using least squares) a new model M1*.
>
> If #(S1*) is less than t, randomly select a new subset S2 and repeat the above process. If, after some predetermined number of trials, no consensus set with t or more members has been found, either solve the model with the largest consensus set found, or terminate in failure.

There are two obvious improvements to the above algorithm: first, if there is a problem-related rationale for selecting points to form the S's, use a deterministic selection process instead of the random one; second, once a suitable consensus set S* has been found and a model M* instantiated, add any new points from P that are consistent with M* to S* and compute a new model on the basis of this larger set.

The RANSAC paradigm contains three unspecified parameters: (1) the error tolerance used to determine whether or not a point is compatible with a model, (2) the number of subsets to try, and (3) the threshold t, which is the number of compatible points used to imply that the correct model has been found. We discuss methods for computing reasonable values for these parameters in the following subsections.

Error Tolerance For Establishing Datum/Model Compatibility

The deviation of a datum from a model is a function of the error associated with the datum and the error associated with the model (which, in part, is a function of the errors associated with the data used to instantiate the model). If the model is a simple function of the data points, it may be practical to establish reasonable bounds on error tolerance analytically. However, this straightforward approach is often unworkable: for such cases it is generally possible to estimate bounds on error tolerance experimentally. Sample deviations can be produced by perturbing the data, computing the model, and measuring the implied errors. The error tolerance could then be set at one or two standard deviations beyond the measured average error.

The expected deviation of datum from an assumed model is generally a function of the datum and, therefore, the error tolerance should be different for each datum. However, the variation in error tolerances is usually relatively small compared to the size of a gross error. Thus, a single error tolerance for all data is often sufficient.

The Maximum Number of Attempts to Find a Consensus Set

The decision to stop selecting new subsets of P can be based upon the expected number of trials k

required to select a subset of n good data points. Let w be the probability that any selected data point is within the error tolerance of the model. Then we have:

$$E(k) = b + 2*(1-b)*b + 3*(1-b)^2*b \ \ldots + i*(1-b)^{i-1}*b + \ldots$$

$$E(k) = b*[1 + 2*a + 3*a^2 \ \ldots + i*a^{i-1} + \ldots]$$

where E(k) is the expected value of k, $b = w^n$, and $a = (1-b)$.

An identity for the sum of a geometric series is:

$$a/(1-a) = a + a^2 + a^3 \ \ldots + a^i + \ldots$$

Differentiating the above identity with respect to a, we have:

$$1/(1-a)^2 = 1 + 2*a + 3*a^2 \ \ldots + i*a^{i-1} + \ldots$$

Thus:

$$E(k) = 1/b = w^{-n}$$

The following is a tabulation of some values of E(k) for corresponding values of n and w:

w	n = 1	2	3	4	5	6
.9	1.1	1.2	1.4	1.5	1.7	1.9
.8	1.3	1.6	2.0	2.4	3.0	3.8
.7	1.4	2.0	2.9	4.2	5.9	8.5
.6	1.7	2.8	4.6	7.7	13	21
.5	2.0	4.0	8.0	16	32	64
.4	2.5	6.3	16	39	98	244
.3	3.3	11	37	123	412	
.2	5.0	25	125	625		

In general, we would probably want to exceed E(k) trials by one or two standard deviations before we give up. We note that the standard deviation of k, SD(k), is given by:

$$SD(k) = \text{sqrt}[E(k^2) - E(k)^2] \quad .$$

Then:

$$E(k^2) = \text{SIGMA(i)}: \{b*i^2*a^{i-1}\}$$

$$= \text{SIGMA(i)}: \{b*i*(i-1)*a^{i-1}\} - \text{SIGMA(i)}: \{b*i*a^{i-1}\}$$

but (using the geometric series identity and two differentiations):

$$2a/(1-a)^3 = \text{SIGMA(i)}: \{i*(i-1)*a^{i-1}\}$$

thus:

$$E(k^2) = (2-b)/(b^2)$$

and:

$$SD(k) = [\text{sqrt}(1 - w^n)]*(1/w^n)$$

We note that generally SD(k) will be approximately equal to E(k): thus, for example, if (w = .5) and (n = 4), then E(k) = 16 and SD(k) = 15.5. This means that we might want to try two or three times the expected number of random selections implied by k (as tabulated above) to obtain a consensus set of more than t members.

From a slightly different point of view, if we want to ensure with probability z that at least one of our random selections is an error-free set of n data points, then we must expect to make at least k selections (n data points per selection), where:

$$(1-b)^k = (1-z)$$

$$k = [\log(1-z)]/[\log(1-b)]$$

For example, if (w = .5) and (n = 4), then (b = 1/16). To obtain a 90% assurance of making at least one error-free selection,

$$k = \log(.1)/\log(15/16) = 35.7$$

A Lower Bound On the Size of an Acceptable Consensus Set

The threshold t, an unspecified parameter in the formal statement of the RANSAC paradigm, is used as the basis for determining that an n-subset of P has been found that implies a sufficiently large consensus set to permit the algorithm to terminate. Thus, t must be chosen large enough to satisfy two purposes: that the correct model has been found for the data; and that a sufficient number of mutually consistent points have been found to satisfy the needs of the final smoothing procedure (which computes improved estimates for the model parameters).

To ensure against the possibility of the final consensus set being compatible with an incorrect model, and assuming that y is the probability that any given data point is within the error tolerance of an incorrect model, we would like y^{t-n} to be very small. While there is no general way of precisely determining y, it is certainly reasonable to assume that it is less than w (w is the a priori probability that a given data point is within the error tolerance of the correct model). Assuming $y < .5$, a value of t-n equal to 5 will provide a better than 95% probability that compatibility with an incorrect model will not occur.

To satisfy the needs of the final smoothing procedure, the particular procedure to be employed must be specified; if least-squares smoothing is to

be used, there are many situations where formal methods can be invoked to determine the number of points required to produce a desired precision (e.g., see Sorenson [1970]).

Example

Let us apply RANSAC to the example described in Figure 1. A value of w (the probability that any selected data point is within the error tolerance of the model) equal to .95 is consistent with the data, and a tolerance (to establish datum/model compatibility) of .8 units was supplied as part of the problem statement. We will accept the RANSAC-supplied model without external smoothing of the final consensus set: thus, we would like to obtain a consensus set of at least seven data points before terminating our search. Since there are only seven data points total and one of these points is a gross error, it is obvious that we will not find a consensus set of the desired size, and so we will terminate with the largest set we are able to find. The theory presented earlier indicates that if we take two data points at a time, compute the line through them and measure the deviations of the remaining points from this line, we should expect to find a suitable consensus set within two or three trials; however, because of the limited amount of data, we might be willing to try all 21 combinations to find the largest consensus set. In either case, we easily find the consensus set containing the six valid data points and the line that they imply.

THE LOCATION DETERMINATION PROBLEM (LDP)

A core problem in image analysis is that of establishing a correspondence between the elements of two representations of a given scene. One variation of this problem, especially important in cartography, is to determine the location in space from which an image or photograph was obtained by recognizing a set of landmarks ("control points") appearing in the image (this is variously called the problem of determining the elements of exterior camera orientation, or the camera calibration problem, or the image-to-data-base-correspondence problem). It is routinely solved using a least-squares technique (e.g. see Wolf [1974] or Keller [1966]) with a human operator interactively establishing the association between image points and the three-dimensional coordinates of the corresponding landmarks. However, in a fully automated system where the correspondences must be based on the decisions of marginally competent feature detectors, least squares is often incapable of dealing with the gross errors that may result: this consideration, discussed at length in the preceding section, is illustrated for the Location Determination Problem in an example to be presented later (see the section on experimental results).

In this section we present a new solution to the Location Determination Problem (LDP) based on the RANSAC paradigm, which is unique in its ability to tolerate gross errors in the input data. We will first examine the conditions under which a solution to the LDP is possible and describe new results concerning this question: we then present a complete description of the RANSAC-based algorithm, and finally, describe experimental results obtained through use of the algorithm.

We formally define the LDP as follows:

> Given a set of m landmarks, whose 3-D coordinates are known in some coordinate frame, and given an image in which some subset of the m landmarks is visible, determine the location (relative to the coordinate system of the landmarks) from which the image was obtained.

We will initially assume that we know the correspondences between n image points and landmarks; later we consider the situation in which some of these correspondences are invalid. We will also assume that both the principal point in the image plane (where the optical axis of the camera pierces the image plane) and the focal length of the imaging system are known; thus (see Figure 2) we can easily compute the angle to any pair of landmarks from the Center of Perspective (CP). Finally, we assume that the camera resides outside and "above" a convex hull enclosing the control points.

We will later demonstrate (Appendix A) that if we can compute the lengths of the rays from the CP to three of the landmarks, then we can directly solve for the location of the CP (and the orientation of the image plane if desired). Thus, an equivalent, but mathematically more concise statement of the LDP, is:

> Given the relative spatial locations of n control points, and given the angle to every pair of control points from an additional point called the Center of Perspective (CP), find the lengths of the line segments ("legs") joining the CP to each of the control points. We call this the "perspective-n-point" problem (PnP).

In order to apply the RANSAC paradigm, we wish to determine the smallest value of n for which it is possible to solve the PnP problem.

Solution of the Perspective-N-Point Problem

The P1P problem (n = 1) provides no constraining information, and thus an infinity of solutions is possible. The P2P problem (n = 2), illustrated in Figure 3, also admits an infinity of solutions; the CP can reside anywhere on a circle of diameter Rab/sin(Oab), rotated in space about the chord (line) joining the two control points A and B.

The P3P problem (n = 3) requires that we determine the lengths of the three legs of a tetrahedron, given the base dimensions and the face angles of the opposing trihedral angle (see Figure 4). The solution to this problem is implied by the three equations [A*]:

$$(Rab)^2 = a^2 + b^2 - 2*a*b*[Cos(Oab)]$$

$$(Rac)^2 = a^2 + c^2 - 2*a*c*[Cos(Oac)] \qquad [A^*]$$

$$(Rbc)^2 = b^2 + c^2 - 2*b*c*[Cos(Obc)]$$

It is known that n independent polynomial equations, in n unknowns, can have no more solutions than the product of their respective degrees. Thus, the system A* can have a maximum of eight solutions. However, because every term in the system A* is either a constant, or of second degree, for every real positive solution there is a geometrically isomorphic negative solution. Thus, there are at most four positive solutions to A*, and in Figure 5 we show an example demonstrating that the upper bound of four solutions is attainable.

In Appendix A we derive an explicit algebraic solution for the system A*. This is accomplished by reducing A* to a biquadratic (quartic) polynomial in one unknown representing the ratio of two legs of the tetrahedron, and then directly solving this equation (we also present a very simple iterative method for obtaining the solutions from the given problem data).

For the case n = 4, when all four control points lie in a common plane (not containing the CP, and such that no more than two of the control points lie on any single line), we provide a technique, in Appendix B, that will always produce a unique solution. Surprisingly, when all four control points do not lie in the same plane, a unique solution cannot always be assured: an example, presented in Figure 6, shows that at least two solutions are possible for the P4P problem with the control points in "general position."

To solve for the location of the CP in the case of four nonplanar control points, we can use the algorithm presented in Appendix A on two distinct subsets of the control points taken three at a time: the solution(s) common to both subsets locate the CP to within the ambiguity inherent in the given information.

The approach used to construct the example shown in Figure 6 can be extended to any number of additional points. It is based on the principal depicted in Figure 3: if the CP and any number of control points lie on the same circle, then the angle between any pair of control points and the CP will be independent of the location on the circle of the CP (and hence the location of the CP cannot be determined). Thus, we are able to construct the example shown in Figure 7, in which five control points in general position imply two solutions to the P5P problem. While the same technique will work for six or more control points, four or more of these points must now lie in same plane and are thus no longer in general position.

To prove that six (or more) control points in general position will always produce a unique solution to the P6P problem, we note that for this case we can always solve for the 12 coefficients of the 3 x 4 matrix T that specifies the mapping (in homogeneous coordinates) from three space to two space: each of the six correspondences provides three new equations and introduces one additional unknown (the homogeneous coordinate scale factor). Thus, for six control points, we have 18 linear equations to solve for the 18 unknowns (actually, it can be shown that, at most, 17 of the unknowns are independent). Given the transformation matrix T, we can construct an additional (synthetic) control point lying in a common plane with three of the given control points and compute its location in the image plane: the technique described in Appendix B can now be used to find a unique solution.

IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

The RANSAC/LD Algorithm

The RANSAC/LD algorithm accepts as input the following data:

(1) A list L of m 6-tuples--each 6-tuple containing the 3-D spatial coordinates of a control point, its corresponding 2-D image plane coordinates, and an optional number giving the expected error (in pixels) of the given location in the image plane.

(2) The focal length of the imaging system and the image plane coordinates of the principal point.

(3) The probability (1-w) that a 6-tuple contains a gross mismatch.

(4) A "confidence" number G which is used to set the internal thresholds for acceptance of intermediate results contributing to a solution. A confidence number of one forces very conservative behavior on the algorithm; a confidence number of zero will call almost anything a valid solution.

The RANSAC/LD algorithm produces as output the following information:

(1) The 3-D spatial coordinates of the lense center (i.e., the Center of Perspective), and an estimate of the corresponding error.

(2) The spatial orientation of the image plane.

The RANSAC/LD algorithm operates as follows:

(1) Three 6-tuples are selected from list L by a quasi-random method that ensures a reasonable spatial distribution for the corresponding control points. This initial selection is called S1.

(2) The CP (called CP1) corresponding to selection S1 is determined using the

closed-form solution provided in Appendix A: multiple solutions are treated as if they were obtained from separate selections in the following steps.

(3) The error in the derived location of CP1 is estimated by perturbing the input coordinates (either by the amount specified in the 6-tuples or by a default value of one pixel) and recomputing the effect this would have on the location of the CP1.

(4) Given the error estimate for the CP1, we use the technique described in Bolles [1978] to determine error ellipses (dimensions based upon the supplied confidence number) in the image plane for each of the control points specified in list L: if the associated image coordinates reside within the corresponding error ellipse, then the 6-tuple is appended to the consensus set S1/CP1.

(5) If the size of S1/CP1 equals or exceeds some threshold value t (nominally equal to a value between 7 and mw), then the consensus set S1/CP1 is supplied to a least-squares routine (see Bolles [1978] or Gennery [1975]) for final determination of the CP location and image plane orientation.* Otherwise, the above steps are repeated with a new random selection S2, S3, ...

(6) If the number of iterations of the above steps exceeds $k = \lceil \log(1-G) \rceil / \lceil \log(1-w^3) \rceil$, then the largest consensus set found so far is used to compute the final solution (or we terminate in failure if this largest consensus set contains fewer than six members).

Experimental Results

To demonstrate the validity of our theoretical results, we performed three experiments. In the first experiment we found a specific Location Determination Problem in which the common least-squares pruning heuristic failed, and showed that RANSAC successfully solved this problem. In the second experiment, we applied RANSAC to fifty synthetic problems in order to check the reliability of the approach over a wide range of parameter values. In the third experiment we used standard feature detection techniques to locate landmarks in an aerial image and then used RANSAC to determine the position and orientation of the camera.

A Location Determination Problem Example of a Least-Squares Pruning Error

The LDP in this experiment was based upon 20 landmarks and their locations in an image. Five of the twenty correspondences were gross errors: that is, their given locations in the image were further than 10 pixels from their actual locations. The image locations for the "good" correspondences were normally distributed about their actual locations with a standard deviation of one pixel.

The heuristic to prune gross errors was the following:

* Use all of the correspondences to instantiate a model.
* On the basis of that model, delete the correspondence that has the largest deviation from its predicted image location.
* Instantiate a new model without that correspondence.
* If the new model implies a normalized error for the deleted correspondence that is larger than three standard deviations, assume that it is a gross error, leave it out, and continue deleting correspondences. Otherwise, assume that it is a good correspondence and return the model that included it as the solution to the problem.

This heuristic successfully deleted two of the gross errors; but after deleting a third, it decided that the new model did not imply a significantly large error, so it returned a solution based upon eighteen correspondences, three of which were gross errors.

When RANSAC was applied to this problem, it located the correct solution on the second triple of selected points. The final consensus set contained all of the good correspondences and none of the gross errors.

Fifty Synthetic Location Determination Problems

In this experiment we applied RANSAC to fifty synthetic LDPs. Each problem was based upon thirty landmark-to-image correspondences. A range of probabilities were used to determine the number of gross errors in the problems: the image location of a gross error was at least 10 pixels from its actual location. The location of a good correspondence was distributed about its actual location with a normal distribution having a standard deviation of one pixel. Two different camera positions were used--one looking straight down on the landmarks and one looking at them from an oblique angle. The RANSAC algorithm described earlier in this section was applied to these problems; however, the simple iterative technique described in Appendix A was used to locate solutions to the P3P problems in place of the closed form method also described in that appendix, and a second least-squares fit was used to extend the final consensus set (as suggested in second section of this paper). Table 1 summarizes the results for ten typical problems (RANSAC successfully avoided including a gross error in its

---

* An alternative to least squares would be to average the parameters computed from random triples in the consensus set that fall within (say) the center 50% of the associated histogram.

final consensus set in all of the problems): in five of these problems the probability of a good correspondence was 0.8 and in the other five problems it was 0.6. The execution time for the current program is approximately one second for each camera position considered.

No. of Good Corresp.	No. of Corresp. in Final Consensus Set	No. of Triples Considered	No. of Camera Positions Considered
	w = .8		
22	19	6	10
23	23	1	3
19	19	2	3
25	25	1	2
24	23	3	8
	w = .6		
21	20	11	21
17	17	1	1
17	16	6	8
18	16	9	21
21	18	9	15

TABLE 1

A "Real" Location Determination Problem

Cross-correlation was used to locate 25 landmarks in an aerial image taken from approximately 4,000 feet with a 6-inch lens. The image was digitized on a grid of 2,000 by 2,000 pixels which implies a ground resolution of approximately two feet per pixel. Three gross errors were made by the correlation feature detector. When RANSAC was applied to this problem, it located a consensus set of 17 on the first triple selected and then extended that set to include all 22 good correspondences after the initial least-squares fit. The final standard deviations about the camera parameters were as follows:

X: 0.1 feet	Heading: .01 degrees
Y: 6.4 feet	Pitch: .10 degrees
Z: 2.1 feet	Roll: .12 degrees

CONCLUDING COMMENTS

In this paper we have introduced a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and thus is ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors.

A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. Most of the results we present concerning solution techniques and the geometry of the LDP problem are either new or not generally known. The current photogrammetric literature offers no analytic solution, other than variants of least squares and the Church method, for solving the perspective-n-point problems. The Church method, which provides an iterative solution for the P3P problem, is presented (see Church [1945] or Wolf [1974]) without any indication that more than one physically real solution is possible: there is certainly no indication that anyone realizes that physically real multiple solutions are possible for more than three control points in general position. (It should be noted that because the multiple solutions can be arbitrarily close together, even when an iterative technique is initialized to a value close to the correct solution, there is no assurance that it will converge to the desired value).

In the section on the LDP problem (and associated appendices) we have completely characterized the P3P problem and provided a closed-form solution. We have shown that multiple physically real solutions can exist for the P4P and P5P problems, but also demonstrated that a unique solution is assured when four of the control points reside on a common plane (solution techniques are provided for each of these cases). The issue of determining the maximum number of solutions possible for the P4P and P5P problems remains open, but we have shown that a unique solution exists for the P6P problem when the control points are in general position.

REFERENCES

1. R. C. Bolles, L. H. Quam, M. A. Fischler, and H. C. Wolf, "Automatic Refinement Of Image-To-Data Base Correspondences " in Proceedings: Image Understanding Workshop, (November 1978).

2. E. Church, "Revised Geometry of the Aerial Photograph - Bulletin of Aerial Photogrammetry No. 15," Syracuse University (1945).

3. S. D. Conte, Elementary Numerical Analysis (McGraw Hill, 1965).

4. E. Dehn, Algebraic Equations (Dover, 1960).

5. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis (Wiley-Interscience, 1973).

6. D. B. Gennery, "Least-Squares Stereo-Camera Calibration," Stanford Artificial Intelligence Project internal memo (1975).

7. M. Keller and G. C. Tewinkel, "Space Resection in Photogrammetry," ESSA Technical Report C&GS 32, U.S. Coast and Geodetic Survey (1966).

8. D. P. Rogers and J. A. Adams, Mathematical Elements for Computer Graphics (McGraw Hill, 1976).

9. H. W. Sorenson, "Least-squares estimation: from Gauss to Kalman," IEEE Spectrum (July 1970).

10. P. R. Wolf, Elements of Photogrammetry (McGraw Hill, 1974).

11. C. R. Wylie, Jr., Introduction to Projective Geometry (McGraw-Hill, 1970).

Appendix A

AN ANALYTIC SOLUTION FOR THE PERSPECTIVE-3-POINT PROBLEM

In the main body of this paper, we have established that P3P problems can have as many as four solutions. In this appendix, we will derive a closed form expression for obtaining these solutions. Our approach involves three steps: we first find the lengths of the legs of the ("perspective") tetrahedron given the base (defined by the three control points) and the face angles of the opposing trihedral angle (the three angles to the three pairs of control points as viewed from the CP). We next locate the CP with respect to the 3-D reference frame in which the control points were originally specified; and finally, compute the orientation of the image plane with respect to the reference frame.

A Solution for the Perspective Tetrahedron (see Figure 4)

Given the lengths of the three sides of the base of a tetrahedron (Rab,Rac,Rbc), and given the corresponding face angles of the opposing trihedral angle (Oab,Oac,Obc). find the lengths of the three remaining sides of the tetrahedron (a,b,c).

A solution to the above problem can be obtained by simultaneously solving the system of equations

$$[A1] \quad (Rab)^2 = a^2 + b^2 - 2*a*b*\cos(Oab)$$

$$[A2] \quad (Rac)^2 = a^2 + c^2 - 2*a*c*\cos(Oac)$$

$$[A3] \quad (Rbc)^2 = b^2 + c^2 - 2*b*c*\cos(Obc)$$

We now proceed as follows:

$$[A4] \quad \text{Let } b = x*a \text{ and } c = y*a$$

$$[A5] \quad (Rac)^2 = a^2 + (y^2)*(a^2) - 2*(a^2)*y*\cos(Oac)$$

$$[A6] \quad (Rab)^2 = a^2 + (x^2)*(a^2) - 2*(a^2)*x*\cos(Oab)$$

$$[A7] \quad (Rbc)^2 = (x^2)*(a^2) + (y^2)*(a^2) - 2*(a^2)*x*y*\cos(Obc)$$

from [A5] and [A7]

$$[A8] \quad ((Rbc)^2)*[\ 1 + (y^2) - 2*y*\cos(Oac)\ ] = ((Rac)^2)*[\ (x^2) + (y^2) - 2*x*y*\cos(Obc)\ ]$$

from [A6] and [A7]

$$[A9] \quad ((Rbc)^2)*[\ 1 + (x^2) - 2*x*\cos(Oab)\ ] = ((Rab)^2)*[\ (x^2) + (y^2) - 2*x*y*\cos(Obc)\ ]$$

$$[A10] \quad \text{Let } \frac{(Rbc)^2}{(Rac)^2} = K1 \quad \text{and} \quad \frac{(Rbc)^2}{(Rab)^2} = K2$$

From [A8] and [A9]

$$[A11] \quad 0 = (y^2)*[1-K1] + 2*y*[K1*\cos(Oac) - x*\cos(Obc)] + [(x^2) - K1]$$

From [A9] and [A10]

$$[A12] \quad 0 = (y^2) + 2*y*[-x*\cos(Obc)] + [(x^2)*(1-K2) + 2*x*K2*\cos(Oab) - K2]$$

Equations [A11] and [A12] have the form:

$$[A13] \quad 0 = m*(y^2) + p*y + q$$

$$[A14] \quad 0 = m'*(y^2) + p'*y + q'$$

Multiplying [A13] and [A14] by m' and m respectively, and subtracting:

$$[A15] \quad 0 = [p*m' - p'*m]*y + [m'*q - m*q']$$

Multiplying [A13] and [A14] by q' and q respectively, subtracting, and dividing by y:

$$0 = [m'*q - m*q']*(y^2) + [p'*q - p*q']*y$$

$$[A16] \quad 0 = [m'*q - m*q']*y + [p'*q - p*q']$$

Assuming m'*q is not equal (#) to m*q', that is

$$[(x^2)-K1] \# [(x^2)*(1-K1)*(1-K2) + 2*x*K2*(1-K1)*\cos(0ab) - (1-K1)*K2]$$

then [A15] and [A16] are equivalent to [A13] and [A14].

We now multiply [A15] by [m'*q - m*q'], and multiply [A16] by [p*m' - p'*m], and subtract to obtain:

$$[A17] \quad 0 = (m'*q - m*q')^2 - [p*m' - p'*m]*[p'*q - p*q']$$

Expanding [A17] and grouping terms we obtain a biquadratic (quartic) polynomial in x:

$$[A18] \quad 0 = G4*(x^4) + G3*(x^3) + G2*(x^2) + G1*(x) + G0$$

where

$$[A19] \quad G4 = (K1*K2 - K1 - K2)2 - 4*K1*K2*[\cos(0bc)^2]$$

$$[A20] \quad G3 = 4*[K1*K2-K1-K2]*K2*(1-K1)*\cos(0ab) + 4*K1*\cos(0bc)*[(K1*K2+K2 -K1)*\cos(0ac) + 2*K2*\cos(0ab)*\cos(0bc)]$$

$$[A21] \quad G2 = [2*K2*(1-K1)\cos(0ab)]^2 + 2*[K1*K2+K1-K2]*[K1*K2-K1-K2] + 4*K1*[(K1-K2)*(\cos(0bc)^2) + (1-K2)*K1*(\cos(0ac)^2) - 2*K2*(1+K1)*\cos(0ab)*\cos(0ac)*\cos(0bc)]$$

$$[A22] \quad G1 = 4*(K1*K2+K1-K2)*K2*(1-K1)*\cos(0ab) + 4*K1*[(K1*K2-K1+K2)*\cos(0ac)*\cos(0bc) + 2*K1*K2*\cos(0ab)*(\cos(0ac)^2)]$$

$$[A23] \quad G0 = (K1*K2+K1-K2)^2 - 4*(K1^2)*K2*(\cos(0ac)^2)$$

Roots of [A23] can be found in closed form (see Dehn [1960]), or by iterative techniques (see Conte [1965]).

For each positive real root of [A18], we determine a single positive real value for each of the sides "a" and "b." From [A6] we have:

$$[24] \quad a = \frac{Rab}{SQRT[(x^2) - 2*x*\cos(0ab) + 1]}$$

and from [A14] we obtain:

$$[A25] \quad b = a*x$$

If m'*q # m*q', then from [A16] we have:

$$[A26] \quad y = \frac{p'*q - p*q'}{m*q' - m'*q}$$

If m'*q = m*q', then [A26] is undefined and we obtain two values of y from [A5]:

$$[A27] \quad y = \cos(0ac) +- SQRT\left[(\cos(0ac))^2 + \frac{(Rac)^2 - (a^2)}{(a^2)}\right]$$

For each real positive value of y, we obtain a value of "c" from [A4]:

$$[A28] \quad c = y*a$$

When values of y are obtained from [A5], rather than [A26], the resulting solutions can be invalid: they must be shown to satisfy [A3] before they are accepted.

It should be noted that because each root of [A18] can conceivably lead to two distinct solutions, the existence of the biquadratic, by itself, does not imply a maximum of four solutions to the P3P problem: some additional argument, such as the one given in the main body of this paper, is necessary to establish the upper bound of four solutions.

Example

For the perspective tetrahedron shown in Figure 5, we have the following parameters:

$$Rab = Rac = Rbc = 2*SQRT(3)$$

$$\cos(0ab) = \cos(0ac) = \cos(0bc) = \frac{(a^2) + (b^2) - (Rab)^2}{2*a*b} = \frac{20}{32}$$

Substituting these values into equations [A19] through [A23], we obtain the coefficients of the biquadratic defined in [A18]:

[-.5625,3.515625,-5.90625,3.515625,-.5625]

The roots of the above equation are:

[1,1,4,0.25]

For each root we have:

ROOT	a	b	y	c
1	4	4	1	4
1	4	4	.25	1
4	1	4	4	4
.25	4	1	1	4

An Iterative Solution for the Perspective Tetrahedron (see Figure 8)

A simple way to locate solutions to P3P problems, which is sometimes an adequate substitute

for the more involved procedure described in the preceding subsection, is to slide one vertex of the control-point triangle down its leg of the tetrahedron and look for positions of the triangle that permit the other two vertices to lie on their respective legs. If vertex A is at a distance "a" from L (L is the center of perspective), the lengths of the sides AB and AC restrict the triangle to four possible positions. Given the angle between legs LA and LB, compute the distance of point A from the line LB and then compute points B1 and B2 on LB that are at the proper distance from A to insert a line segment of length AB. Similarly, we compute (at most) two locations for C on its leg. Thus, given a position for A, we have found (at most) four positions for a triangle that has one side of length AB and one of length AC. The lengths of the third sides (BC) of the four triangles vary (non-linearly) as point A is moved down its leg. Solutions to the problem can be obtained by iteratively repositioning A to imply a third side of the required length.

Computing the 3-D Location of the Center of Perspective (see Figure 9)

Given the three-dimensional locations of the three control points of a perspective tetrahedron, and the lengths of the three legs, the 3-D location of the center of perspective can be computed as follows:

(1) Construct a plane P1 that is normal to AB and passes through the center of perspective, L. This plane can be constructed without knowing the position of L, which is what we are trying to compute. Consider the face of the tetrahedron that contains vertices A, B, and L. Knowing the lengths of sides LA, LB and AB, we can use the law of cosines to find the angle LAB, and then the projection QA of LA on AB. (Note that angle LQA is a right angle, and the point Q is that point on line AB that is closest to L). Construct a plane normal to AB passing through Q; this plane also passes through L.

(2) Similarly construct a plane P2 that is normal to AC and passes through L.

(3) Construct the plane P3 defined by the three points A, B, and C.

(4) Intersect planes P1, P2, and P3. By construction, the point of intersection R is the point on P3 that is closest to L.

(5) Compute the length of the line AR and use that in conjunction with the length of LA to compute the length of the line RL, which is the distance of L from the plane P3.

(6) Compute the cross product of vectors AB and AC to form a vector perpendicular to P3. Then scale that vector by the length of RL and add it to R to get the 3-D location of the center of perspective L.

If the focal length of the camera and the principal point in the image plane are known, it is possible to compute the orientation of the image plane with respect to the world coordinate system; that is, the location of the origin and the orientation of the image plane coordinate system with respect to the 3-D reference frame. This can be done as follows:

(1) Compute the 3-D reference frame coordinates of the center of perspective (as described above).

(2) Compute the 3-D image locations for the three control points: Since we know the 3-D coordinates of the CP and control points, we can compute the 3-D coordinates of the three rays between the CP and the control points. Knowing the focal length of the imaging system, we can compute, and subtract from each ray, the distance from the CP to the image plane along the ray.

(3) Compute the equation of the plane containing the image using three of the image points found in step (2). The normal to this plane, passing through the CP, gives us the origin of the image plane coordinate system (i.e., the 3-D location of the principal point), and the Z axis of this system.

(4) The 3-D orientation of the image plane X and Y axis can now be obtained by computing the 3-D coordinates of a vector from the principal point to any of images of the points found in (2).

Appendix B

AN ANALYTIC SOLUTION FOR THE PERSPECTIVE-4-POINT PROBLEM (with all control points lying in a common plane)

In this appendix, we present an analytic technique for obtaining a unique solution to the P4P problem, when the four given control points all lie in a common plane:

Problem Statement (see Figure 10)

GIVEN: a correspondence between four points lying in a plane in 3-D space (called the object plane), and four pointa lying in a diatinct plane (called the image plane); and given the distance between the center of perspective and the image plane (i.e., the focal length of the imaging system); and also given the principal point in the image plane (i.e., the location, in image plane coordinates, of the point at which the optical axis of the lense pierces the image plane).

FIND: the 3-D location of the Center of Perspective relative to the coordinate system of the object plane.

Notation

* Let the four given image points be labeled {Pi}, and the four corresponding object points {Qi}.
* We will assume that the 2-D Image Plane coordinate system has its origin at the principal point (PPI).
* We will assume that the Object Plane has the equation Z = 0 in the reference coordinate aystem. Standard techniques are available to transform from this coordinate system into a ground reference frame (e.g., see Duda [1973] or Rogers [1976]).
* Homogeneous coordinates will be assumed (e.g., see Wylie [1970]).
* Primed symbols represent transposed structures.

Solution Procedure

a) Compute the 3 x 3 collineation matrix T which mapa points from Object Plane to Image Plane (a procedure for computing T is given later):

(1) $[Pi] = [T]*[Qi]$
where $[Pi] = [ki*xi, ki*yi, ki]'$
$[Qi] = [Xi, Yi, 1]'$

b) The ideal line in the Object Plane, with coordinates $[0.0\ 1]'$, is mapped into the vanishing line in the Image Plane [VLI] by the transformation:

(2) $[VLI] = [inv[T]]'*[0,0,1]'$

c) Determine the distance DI from the origin of the Image Plane (PPI) to the vanishing line $[VLI] = [a1, a2, a3]'$:

(3) $$DI = \left| \frac{a3}{\text{sqrt}[(a1)^2 + (a2)^2)]} \right|$$

d) Solve for the dihedral (tilt) angle 0 between the Image and Object planes:

(4) $$0 = \arctan\left(\frac{f}{DI} \right)$$

where f = focal length

e) The ideal line in the Image Plane with coordinates $[0,0,1]'$ is mapped into the vaniahing line in the Object Plane [VLO] by the transform:

(5) $[VLO] = [T]'*[0,0,1]'$

f) Compute the location of point [PPO] in the Object Plane ([PPO] is the point at which the optical axis of the lense pierces the object plane):

(6) $[PPO] = [inv[T]]'*[0,0,1]'$

g) Compute the distance DO from $[PPO] = [c1, c2, c3]'$ to the vanishing line $[VLO] = [b1.b2, b3]'$ in the Object Plane:

(7) $$DO = \left| \frac{b1*c1 + b2*c2 + b3*c3}{c3*\text{aqrt}[(b1)^2 + (b2)^2]} \right|$$

h) Solve for the "pan" angle $ as the angle between the normal to $[VLO] = [b1, b2, b3]'$ and the X axis in the Object Plane:

(8) $$\$ = \arctan\left(\frac{-b2}{b1} \right)$$

i) Determine XSGN and YSGN:

If a line (parallel to the X axis in the object plane) through [PPO] intersects [VLO] to the right of [PPO], then XSGN = 1 else XSGN = -1. Thus

(9) $$\text{if } \frac{b1*c1 + b2*c2 + b3*c3}{b1*c3} < 0$$

then XSGN = 1 else XSGN = -1

Similarly,

(10) $$\text{if } \frac{b1*c1 + b2*c2 + b3*c3}{b2*c3} < 0$$

then YSGN = 1 elae YSGN = -1

j) Solve for the location of the CP in the object plane coordinate system:

(11) DCP = DO*ain(0)

(12) XCP = XSGN*abs[DCP*Sin(0)*Coa($)] + c1/c3

(13) YCP = YSGN*abs[DCP*Sin(0)*Sin($)] + c2/c3

(14) ZCP = DCP*cos(0)

Note: If [VLI], as determined in (b), has the coordinates [0,0,k], then the image and object planea are parallel (0 = 0). Rather than continuing with the above procedure,

90

we now solve for the desired information using similar triangles and Euclidean geometry.

Computing the Collineation Matrix T

Let:

$$[Q] = \begin{Vmatrix} X1 & Y1 & 1 \\ X2 & Y2 & 1 \\ X3 & Y3 & 1 \end{Vmatrix} = [[Q1]',[Q2]',[Q3]']$$

$$[P] = \begin{Vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{Vmatrix} = [[P1]',[P2]',[P3]']$$

$$[Q4] = [X4,Y4,1]'$$

$$[P4] = [x4,y4,1]'$$

$$[V] = [inv[P]]'*[P4] = [v1,v2,v3]'$$

$$[R] = [inv[Q]]'*[Q4] = [r1,r2,r3]'$$

$$w1 = \frac{v1}{r1} * \frac{r3}{v3}$$

$$w2 = \frac{v2}{r2} * \frac{r3}{v3}$$

$$[w] = \begin{Vmatrix} w1 & 0 & 0 \\ 0 & w2 & 0 \\ 0 & 0 & 1 \end{Vmatrix}$$

Then:

$$[T]' = [INV[Q]]*[W]*[P]$$

Such that:

$$[Pi] = ki*[xi,yi,1] = [T]*[Qi].$$

Example

Given

f = .3048 meters (12 inches)

P1 = (-.071263 .029665) Q1 = (-30, 80)
P2 = (-.053033, -.006379) Q2 = (-100, -20)
P3 = (-.014063. .061579) Q3 = (140, 50)
P4 = (.080120, -.030305) Q4 = (-40, -240)

a) $$[T]' = \begin{Vmatrix} .000212 & .000236 & .000925 \\ -.000368 & .000137 & .000534 \\ -.025404 & .021650 & .843879 \end{Vmatrix}$$

$$[inv[T]]' = \begin{Vmatrix} 1117.14 & -2038.86 & 0.0 \\ 3371.56 & 2302.22 & -5.14991 \\ -51.0636 & -120.442 & 1.31713 \end{Vmatrix}$$

b) [VLI] = [0, -5.14991, 1.31713]'

c) DI = .255758

d) O = .872665 radians (50 degrees)

e) [VLO] = [.000925, .000534, .843880]'

f) [PPO] = [-51.0636, -120.442, 1.31713]'

g) DO = 711.196

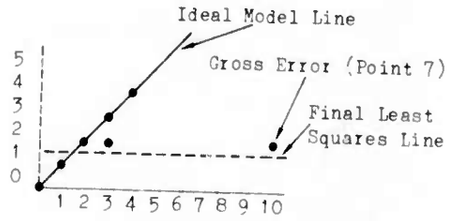
h) $ = -.523599 radians (-30 degrees)

i) XSGN = -1
YSGN = -1

j) DCP = 544.8081

XCP = -400.202
YCP = -300.117
ZCP = 350.196

Problem: Given the set of seven (x, y) pairs shown in the plot, find a best fit line, assuming that no valid datum deviates from this line by more than 0.8 units.



Point	x	y
1	0	0
2	1	1
3	2	2
4	3	2
5	3	3
6	4	4
7	10	2

Comment: Six of the seven points are valid data and can be fit by the solid line. Using Least Squares (and the "throwing out the worst residual" heuristic), we terminate after four iterations with four remaining points, including the gross error at (2, 10) fit by the dashed line.

Successive Least Squares Approximations

Iteration	Data Set	Fitting Line
1	1,2,3,4,5,6,7	1.48 + .16x
2	1,2,3,4,5,7	1.25 + .13x
3	1,2,3,4,7	.96 + .14x
4	2,3,4,7	1.51 + .06x

Computation of Residuals

Point	Iteration 1 Residuals	Iteration 2 Residuals	Iteration 3 Residuals	Iteration 4 Residuals
1	1.48	1.25	.96*	--
2	.64	.38	.10	.57
3	.20	.49	.76	.37
4	.05	.36	.63	.31
5	1.05	1.36*	--	--
6	1.89*	--	--	--
7	1.06	.57	.33	.11

FIGURE 1 FAILURE OF LEAST SQUARES (AND THE "THROWING OUT OF THE WORST RESIDUAL" HEURISTIC), TO DEAL WITH AN ERRONEOUS DATA POINT

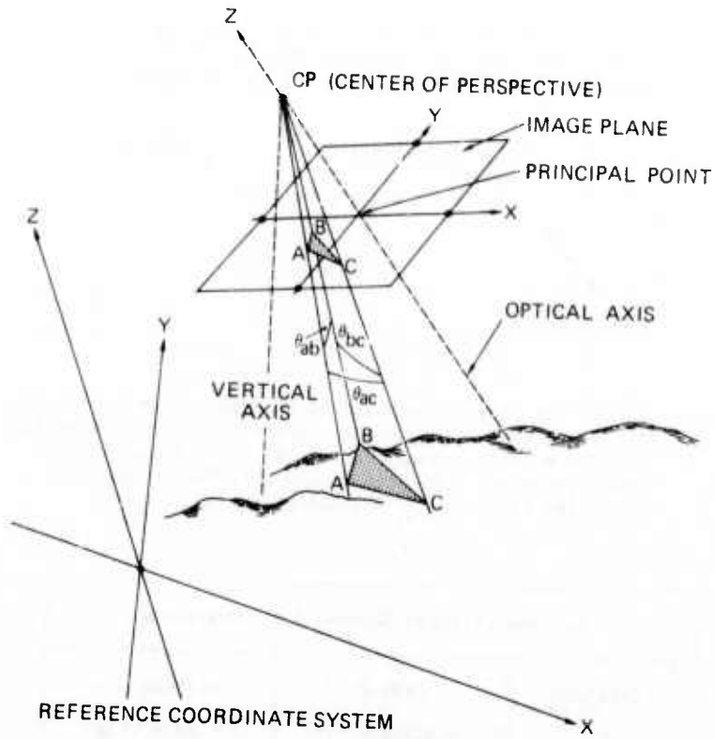
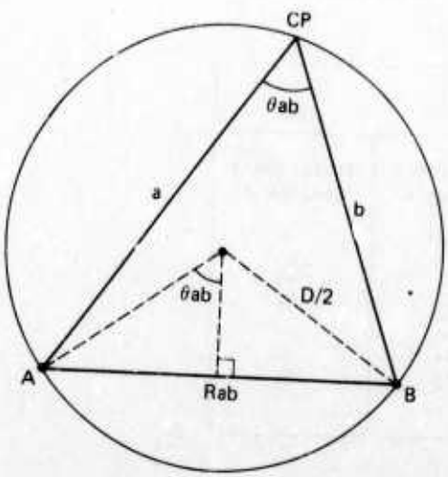


FIGURE 2 GEOMETRY OF THE LOCATION DETERMINATION PROBLEM



$$\text{Sin } \theta ab = \frac{Rab/2}{D/2}$$

$$D = \frac{Rab}{\text{Sin } \theta ab}$$

FIGURE 3 GEOMETRY OF THE P2P PROBLEM

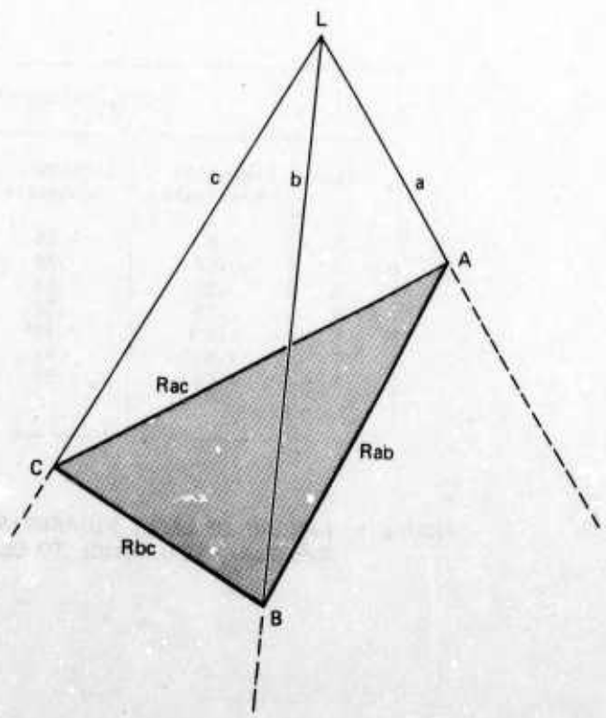
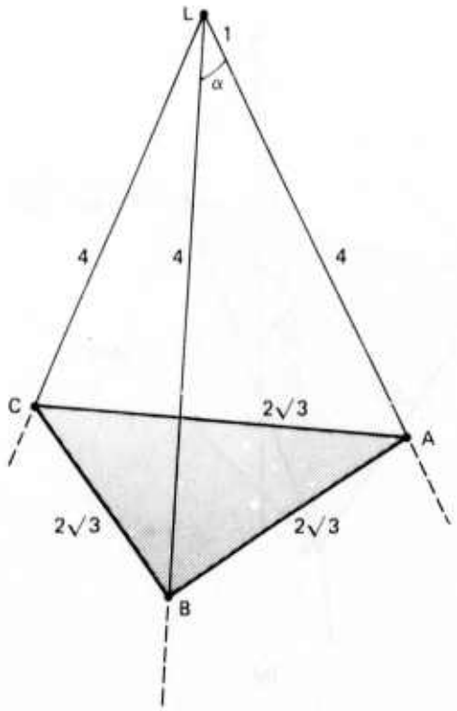
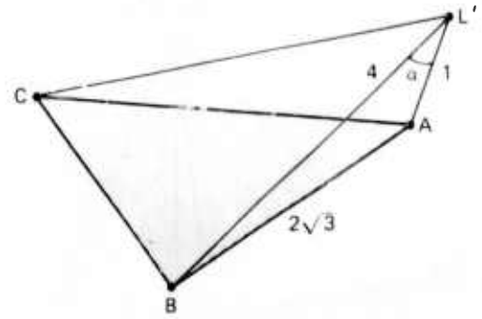


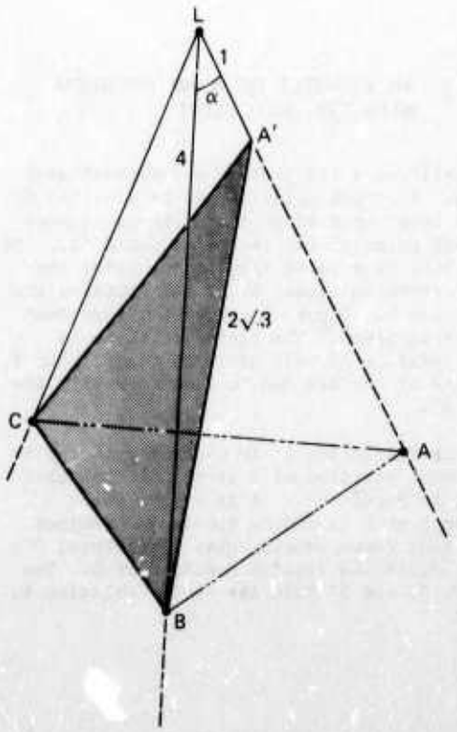
FIGURE 4 GEOMETRY OF THE P3P PROBLEM (L IS THE CENTER OF PERSPECTIVE)



(a)



(b)



(c)

FIGURE 5 AN EXAMPLE SHOWING FOUR DISTINCT SOLUTIONS TO A P3P PROBLEM

Consider the tetrahedron in Figure 5a. The base ABC is an equilateral triangle and the "legs" (i.e., LA, LB, and LC) are all equal. Therefore, the three face angles at L (i.e., <ALB, <ALC, and <BLC) are all equal. By the law of cosines we have:

$$\text{Cos(Alpha)} = 5/8.$$

This tetrahedron defines one solution to a P3P problem. A second solution is shown in Figure 5b. It is obtained from the first by rotating L about BC. It is necessary to verify that the length of L'A can be 1, given the rigid triangle ABC and the angle alpha. From the law of cosines we have:

$$(2*\text{sqrt}(3))^2 = 4^2 + (L'A)^2 - 2*4*(L'A)*(5/8)$$

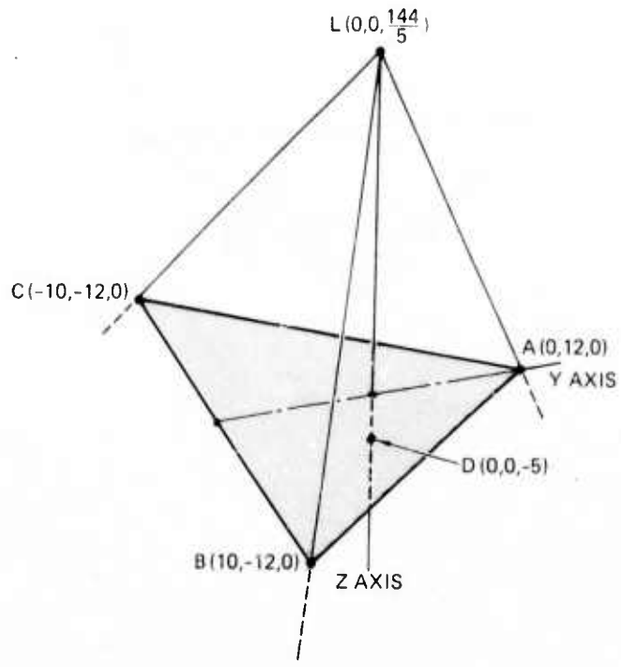
which reduces to:

$$(L'A - 1) * (L'A - 4) = 0.$$

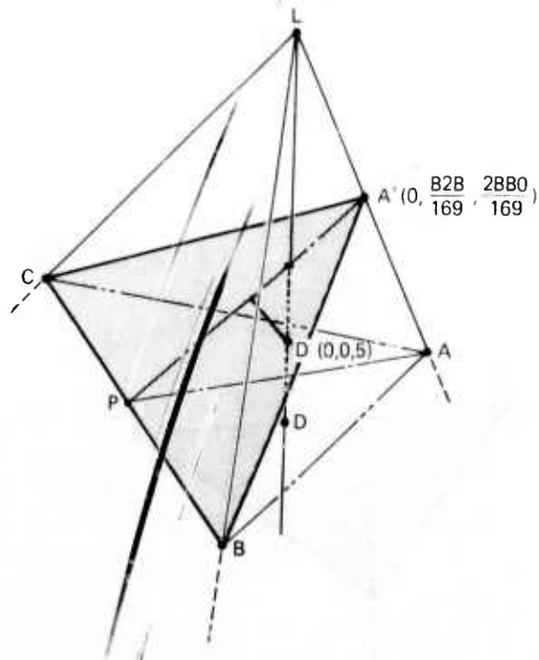
Therefore, L'A can be either 1 or 4. Figure 5a illustrates the L'A = 4 case and Figure 5b illustrates the L'A = 1 case.

Notice that repositioning the base triangle so that its vertices move to different locations on the legs is equivalent to repositioning L. Figure 5c shows the position of the base triangle that corresponds to the second solution.

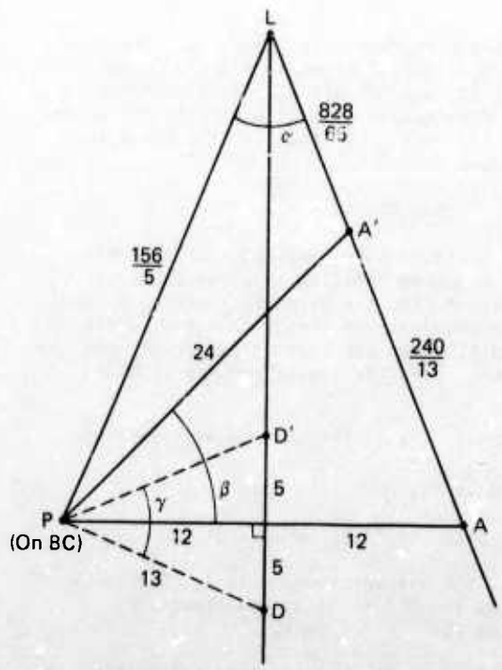
Since the tetrahedron in Figure 5a is threefold rotationally symmetric, two more solutions can be obtained by rotating the triangle about AB and AC.



(a)



(b)



(c)

FIGURE 6 AN EXAMPLE OF A P4P PROBLEM WITH TWO SOLUTIONS

Figure 6a specifies a P4P problem and demonstrates one solution. A second solution can be achieved by rotating the base about BC so that A is positioned at a different point on its leg (see Figure 6b). To verify that this is a valid solution consider the plane X = 0, which is normal to BC and contains the points L, A, and D. Figure 6c shows the important features in this plane. The cosine of alpha is 119/169. A rotation of beta about BC repoaitions A at A'. The law of cosines can be used to verify the position of A'.

To complete this solution it is necessary to verify that the rotated position of D is on LD. Conaider the point D' in Figure 6c. It is at the same distance from P as D is and by the law of cosines we can show that gamma equals beta. Therefore, D', which is on LD, is the rotated poaition of D. The points A', B, C, and D' form the second aolution to the problem.

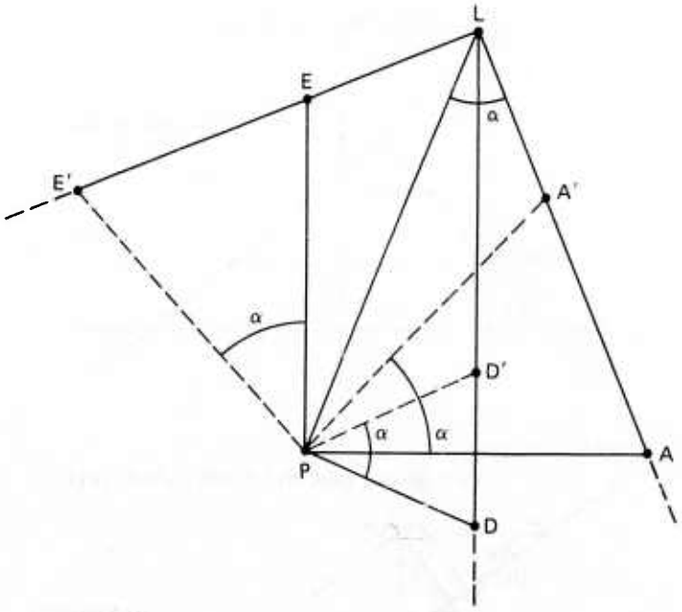
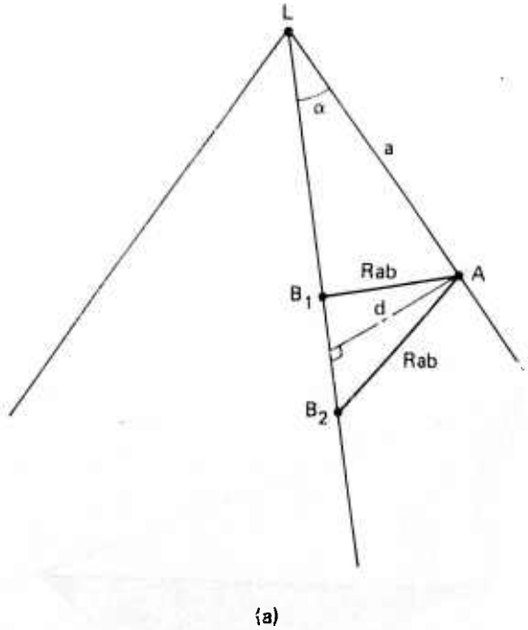


FIGURE 7 AN EXAMPLE OF A P5P PROBLEM WITH TWO SOLUTIONS

This example is the same as the P4P example described in Figure 6 except that a fifth control point, E, has been added. The initial position for E and its rotated position, E', are shown in Figure 7. The points E and E' were constructed to be the mirror images of A' and A about the line LP; therefore, a rotation of alpha about P repositions E at E'. One solution of the P5P problem is formed by points A, B, C, and D (shown in Figure 6a) plus point E. The second solution is formed by points A', B, C, D', and E'. Consequently there are two different positions of L such that all five points lie on their appropriate legs.



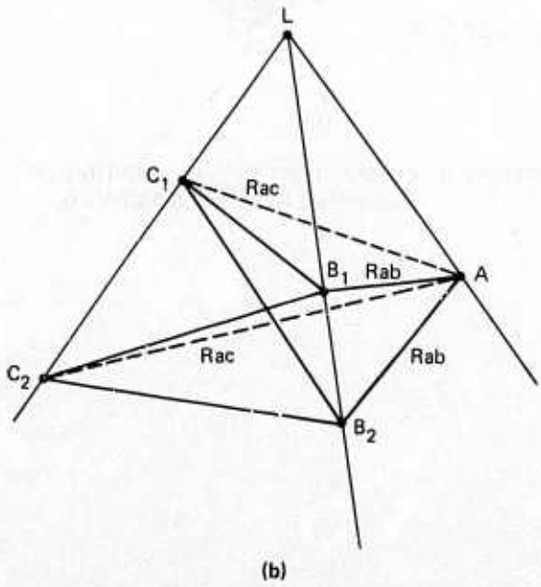


FIGURE 8 GEOMETRY FOR AN ITERATIVE SOLUTION TO THE P3P PROBLEM

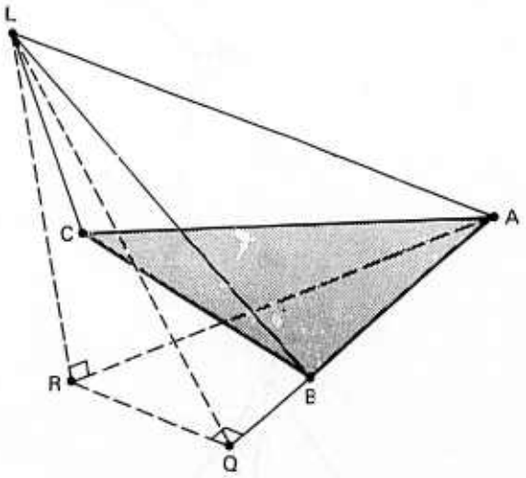
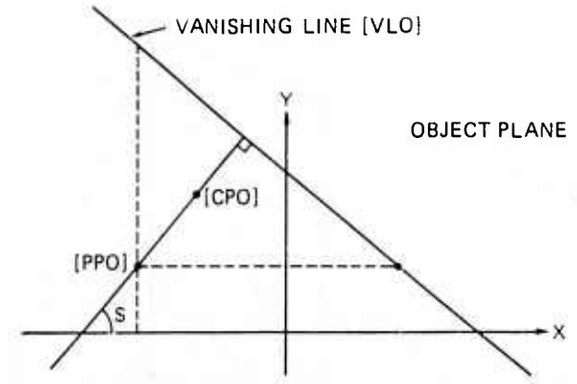


FIGURE 9 COMPUTING THE 3-D LOCATION OF THE CENTER OF PERSPECTIVE (L)



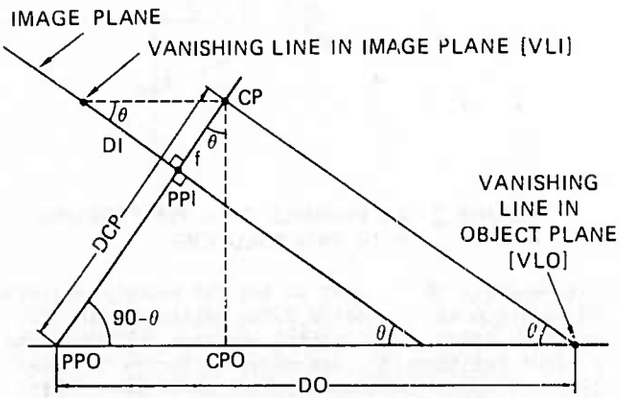


FIGURE 10 GEOMETRY OF THE P4P PROBLEM (WITH ALL CONTROL POINTS LYING IN A COMMON PLANE)

SEMANTIC DESCRIPTION OF AERIAL IMAGES USING STOCHASTIC LABELING

O.D. Faugeras and K.E. Price

Image Processing Institute
University of Southern California
Los Angeles, California 90007

ABSTRACT

This paper discusses the application of stochastic labeling to a general symbolic image description problem. A method used to compute initial likelihoods and compatibilities is described. It was derived from an earlier symbolic matching procedure, but was modified to provide the data needed for application of the labeling method. This labeling procedure differs from simpler ones, in that it maximizes a global criterion at each iteration. This technique is compared to other matching methods and results on two scenes are presented.

INTRODUCTION

The purpose of computer scene analysis is to automatically produce a description of the content of an image similar to one obtained from a skilled human observer. In order to achieve such a goal, a symbolic description of the raw image data must be constructed. This requires the application of many of the now well developed techniques of Image Processing (image bandwidth compression, image restoration and image enhancement), extraction of features such as texture and edges, segmentation of the image into homogeneous regions with respect to one or several properties, and measuring features that characterize these segments (color, brightness, texture, size, and shape) and also relations between these segments (brighter than, larger than, above, below, etc.). The output of this complex sequence of processes is something that does not resemble the original input array of pixels but is much more suitable for high level processing: a symbolic description which is represented as a labelled graph or semantic network. To proceed any further, we must also assume that we have access to another body of knowledge containing a priori information about the expected content of images of a given area. We will not make any assumptions about how this world model has been obtained (manual input or intelligent learning) and will only assume that its representation is the same as the image, i.e., that it is also a semantic network. The process of obtaining a semantic description of a given image can then be viewed as finding the solution of a graph matching problem: either match the image onto the model or match the model onto the image. Thus, this is equivalent to the general graph/subgraph isomorphism problem which is well known to be NP-complete. Practical and useful solutions can nonetheless be found as we will show in this paper.

Matching techniques must be described in terms of performance on a well defined problem. The particular task under study is the analysis of an image of a scene using an approximate specification of the scene which would apply for many different images of the scene (i.e. a model). Both the image and scene are represented by semantic networks, the image description is automatically generated and thus reflects any errors in the segmentation process. The model is specified by the user and contains only the important objects and relations.

A similar problem was attacked by Rubin [1] with a search procedure and a more detailed model. His work has been combined with a relaxation procedure by Smith [2]. The general form of the solution is similar to ours but the scene model is significantly more detailed so that exact comparisons cannot be made.

We will first present the image and model descriptions, which are the input to the matching procedure, then the basic matching technique which we use to compare two objects. Next, the global matching process (labeling) will be described, finally we will discuss the results of applying this procedure.

DESCRIPTION AND BASIC MATCHING TECHNIQUES

This section will first present a description of the symbolic representation used for the scene and image. Then the method by computing initial likelihoods of particular matches is described along with the method to compute compatibilities of pairs of matches.

1. Image And Model Descriptions

Our matching system uses a feature based, symbolic, description of an idealized scene (the model) and of the image of a portion of this scene [3]. The image description is derived automatically from an image and the model is developed by the user through an interactive procedure.

The basic objects used for the image description are the segments of the image

generated both by a general region based image segmentation procedure [4] and by a linear feature extraction procedure [5]. The regions are derived by locating connected areas which are uniform with respect to some feature in the input image (color parameters, texture values etc.). Linear features are defined as long narrow objects which differ from the background on both sides, and are described by as a sequence of straight line segments with some small width. Typically, the images which we use have a total of 100-200 individual segments of both types. The symbolic description is completed by computing various features of the segments and relations between them.

The features used for the symbolic description are those which can be reliably computed from the available data (the input image and the region or line descriptions). They include properties such as average color and texture (currently only simple texture measures), size, position, two-dimensional orientation, and simple shape measures. Also included are various relations between image segments such as adjacency, nearby, and relative positions (above, below, etc.). With all these relations a segment may easily be related to as many as 100 other segments. This description is not intended to be used for reconstruction of the original image, it is meant to capture the important, observable, information contained in the image.

The model description is identical to that used for the image - feature based descriptions of basic region-like and line-like elements including relations between them. Additionally, the basic elements in the model are grouped into more complex objects, associated with generic descriptions, and referred to by actual names. The feature values in the model will not correspond exactly to image values, but are approximations of the likely values, so that one model of a scene can be used with many similar images of the same scene. The relations between elements which are included in the model are the "important" ones, that is, if a relation appears in the model description then it is expected to occur in the image description, but, if no relation occurs in the model then nothing may be said about its appearance in the image (negative relations could be used, such as must not be adjacent, etc.). Similarly, only the important objects are described in the model, thus it is not a complete description of the entire scene. Generally, the model description is smaller than the image description containing 20-30 basic elements.

In summary, the input for the matching procedures is the symbolic descriptions of both the input image and a model of the scene. The model description determines the outcome of the matching operations. The image description is automatically derived and may contain errors - especially where simple objects are broken into several pieces. The model description is incomplete, as it contains only important objects, thus most segments in the image (and objects in the actual scene) will not be described by objects in the model.

2. Basic Matching Technique

The global matching procedure is a stochastic labeling procedure (also called relaxation procedure) which will be explained in detail in the next section. The relaxation technique requires a basic procedure to compare how well a model element agrees with an image segment for its operation. In our previous work in matching pairs of images [2] and in analysis of images using models of the scene [6] we have developed a comparison procedure which can rate the correspondence of an object in one image (or model) with an object in another image (or model). The basic procedure combines differences in all available features and relations to produce a single rating of the quality of the match. The past experiments indicated that this procedure produces reliable, and generally accurate measures for the differences between two objects (i.e. the model based matching performed accurately on a variety of scenes). The problems with the past matching system arose in the use of these results by the global matching procedure, particularly in requirements for ordering the selection of elements to match and the handling of objects which break into several pieces. (This is discussed in more detail in part IV.)

Briefly, the basic matching procedure combines differences in all feature values which are weighted to account for the difference ranges of values (small size differences (1000 pixels) are not as important for large regions but small changes in orientation (0.5 radian) are very significant). Additionally, differences in the number of relations in the model and the number of the same relations between the corresponding image segments are used. All of these components are given a strength (high, medium, low) to control their impact on the final match result (i.e. features known to be marginally useful are given a low strength, and those considered very important are given a high strenth). In the earlier implementation the absolute value of the rating ranged from 0 upward to 10000 or more. These values are converted to the range [0.0, 1.0] by using the reciprocal of the value plus 1.

The stochastic labeling operation uses the matching procedure for two distinct purposes. First, it is necessary to determine the initial likelihood of a particular assignment (i.e. a rating without consideration of neighbors, or in other words use only the unary relations). Second, the compatibility of particular assignments for two objects must be computed (i.e. the rating using the relations between two objects).

The computation of initial likelihoods

cannot use relations between objects since their use depends on assignments of model elements to image segments. Therefore, the initial match is limited to feature values (color, size, shape, etc.). A model element is compared with all the image segments using our basic matching procedure. The best matching segments are kept for further analysis, currently up to thirty are used or up to the point where the worst is 1/10 of the best (whichever given the smaller set). Then the match results are scaled so that they sum to 1, to be treated as probabilities in the stochastic labeling procedure.

Clearly, if feature values, alone, are sufficient to locate correct matches, then the process could stop at this point. But, even though features are sufficient for some well defined objects, they do not locate most correspondences by themselves.

After an initial application of the relaxation procedure several assignments may be made. At this point, the computation of initial likelihoods for an object can, and does, use the relations with assigned elements. This means that initial likelihoods are always computed with all available information, initially only feature values then an increasing number of relations. Therefore successive steps which are using more information can more reliably match the less well defined objects.

The compatibility measure computes the effect of making an assignment for one element on the assignment for another element. The interaction between objects and their assignments is through the relations between them, so that the compatibility measure is based on these relations. Here again, the same basic matching procedure is used to compute how well the relations in the model match the relations in the image. The procedure has been modified so that only the relations between the two model elements and between the potential corresponding image elements are considered.

In some implementations of a stochastic labeling procedure, all the possible compatibility measures are computed once at the beginning, but because of the total number of possibilities which would be required, this can not be done. Since only a small fraction of the total number are ever required, they are computed as needed. Clearly, in one experiment, the same value will be computed several times, but the cost is small.

The compatibility measure is computed using the most likely assignments for the second object. The individual matches are weighted by the likelihood of a particular assignment so that more likely assignments contribute more to the result. The number of likely assignments to be considered is determined by an input parameter. The greater the number of assignments, the greater time the procedure will take. Experiments have indicated that using more than one or two assignments does not contribute much to the final results (see the final section). If an object has a firm assignment (i.e. some segment has been selected as corresponding to a model object) then this assignment is included in the compatibility computation in addition to the number of assignments specified by the parameter described above.

RELAXATION ALGORITHM

This section describes the basic stochastic labeling and the optimization techniques algorithms. The second part of the section presents the variations of the basic procedure which were required for this symbolic matching problem.

1. General Description

Relaxation labeling attempts to efficiently solve a very general problem in Pattern Recognition and Artificial Intelligence: given a set of units U and a set of names N, assign names to units given actual measured features and a world model. There are two broad classes of Relaxation techniques. The first one, called discrete Relaxation, handles the case where, for every unit, we can know if a name is possible or impossible. Discrete Relaxation is then an efficient way to solve the search problem of finding a labeling of the units. Continuous Relaxation handles the cases where we know more than just whether names are possible or impossible, namely a measure of their likelihood.

In the first case, the world model consists of a binary relation $R \subset (UxN)x(UxN)$ that determines whether assigning name n_1 to unit u_1 is compatible with assigning name n_2 to unit u_2. In the second case, this compatibility is given by a positive number $c(u_1,n_1,u_2,n_2)$, which is small if the compatibility is weak and large if it is strong. Function c is defined in general only over a subset S of (UxN)x(UxN). To every unit u_i and name n_k we can thus associate the set $V_{ij}(k)$ of related units u_j such that there exists a name n_ℓ for which (u_1,n_k,u_j,n_ℓ) is in S.

In this paper, we are solely concerned with continuous Relaxation, where for every unit u_i, there is a corresponding probability vector $\vec{p}_i = [p_i(1)..p_i(L_i)]^T$ where $p_i(k)$ $(1 \le k \le L_i)$ measures the probability that unit u_i has the name n_k. The set of all vectors $\vec{p}_i$ is called a stochastic labeling of the set of units U. As proposed in [7,8] we can also define for each unit u_i a compatibility or prediction vector $\vec{q}_i = [q_i(1)...q_i(L_i)]$ that tells us what $\vec{p}_i$ should be, given the probability of assignments $\vec{p}_j$ at neighboring units u_j and the world model embedded in function c. For simplicity we will rewrite $c(u_i,n_k,u_j,n_\ell)$ as $c(i,k,j,\ell)$ in what follows.

As described in [7,8] we can take

$$q_i(k) = \frac{Q_i(k)}{\sum_{\ell=1}^{L_i} Q_i(\ell)} \quad (1)$$

where

$$Q_i(k) = \sum_{u_j \text{ in } V_i(\ell)} \sum_{\ell \text{ in } W_j}^{L_j} c(i,k,j,\ell)\, p_j(\ell) \quad (2)$$

and W_j is a subset of the possible names for unit u_j. In the experiments reported in [7,8] we took $W_j = \{1,\ldots,L_j\}$; but because of the large number of possible names in this task and for the sake of efficiency we took

$$W_j = \{\text{set of n most likely names}\}$$

usually with n = 1. That is, for every neighbor of a unit we considered only the contribution of the most likely name in Eq. (2).

In [7,8] we proposed to use local measures of consistency and ambiguity of the form

$$\alpha \|\vec{p}_i - \vec{q}_i\|_2^2 + (1-\alpha) H_i \quad (3)$$

where $\|\cdot\|_2$ is the usual Euclidean norm, H_i the quadratic entropy

$$H_i = \sum_{\ell=1}^{L_i} p_i(\ell)(1-p_i(\ell)) = 1 - \|\vec{p}_i\|_2^2 \quad (4)$$

and α a weighting factor adjusting the relative importance of consistency versus ambiguity. It was found in [9] that an even better measure is given by the inner product

$$\vec{p}_i \cdot \vec{q}_i \quad (5)$$

The global measure is then an average over the set of units of the local measures. We can thus define

$$C = \sum_{\substack{\text{all units} \\ u_i}} \vec{p}_i \cdot \vec{q}_i \quad (6)$$

as a global criterion over the set of units that measures the consistency and ambiguity of the labeling.

The labeling problem is now equivalent to the following:

$\mathcal{P}$ { given an initial labeling $\vec{p}_i^{(0)}$, find the local maximum of criterion C closest to the original labeling subject to the constraint that vectors $\vec{p}_i$ are probability vectors.

This is typically a constrained optimization problem which as shown in [7,8] can be efficiently solved by using steepest descent techniques. In particular, we can attach to every unit u_i a local gradient vector $\vec{g}_i = \frac{\partial C}{\partial \vec{p}_i}$ and define an iteration scheme as:

$$\vec{p}_i^{(n+1)} = \vec{p}_i^{(n)} + \rho_n P\{\vec{g}_i^{(n)}\} \quad n \geq 0 \quad (7)$$

where ρ_n is a positive number and P a linear projection operator. For a description of P see [7].

It can be easily shown that:

$$g_i(k) = q_i(k) + \sum_{\substack{\text{neighbors} \\ \text{of } u_i}} \frac{1}{D_j} \sum_{\ell \text{ in } W_j} C(j,\ell,i,k) \cdot (p_j(\ell) - \vec{p}_j \cdot \vec{q}_j) \text{ for } k=1,\ldots,L_i \quad (8)$$

where

$$D_j = \sum_{\ell=1}^{L_j} Q_j(\ell) \quad (9)$$

The first term $q_i(k)$ in Eq. (6) corresponds to the simple maximization of the product $\vec{p}_i \cdot \vec{q}_i$ in the global criterion C, whereas the second term corresponds to the coupling between units through the compatibility relations. The algorithm defined by Eq. (5) will allow us to evolve from the initial stochastic labeling toward a less ambiguous and more consistent labeling.

2. Least Commitment Versus Speed, Multiple Matches

The task of matching a model with a symbolic representation of an image obtained by a automatic segmentation procedure presents the important characteristic that matches may not be unique. For example, if a highway in the image has been separated by the segmentation procedure into several disconnected pieces, then each piece is a potential correct match for the node "highway" in the model.

One possible way of handling this problem would be to relax the constraint that vectors $\vec{p}_i$ are probability vectors and interpret them as confidence vectors for which every component may vary between 0 and 1. This would have the benefit of delaying any firm commitment as much as possible, but would also have a general tendency to slow the convergence of the iterative scheme. This is why we looked for an alternative between an increase in speed (and therefore of the probability of making errors) and the principle of least commitment.

We therefore introduced the notion of a Macro-iteration composed of several of the iterations defined by Eq. (7) (Micro-iterations) after which decisions are made to assign names to units based upon the comparison of the components of the vectors $\vec{p}_i$ to a threshold (usually 80%). If one component is larger than the threshold then it is set equal to 1 and unit u_i is considered to be assigned the corresponding name n_k.

The process is then reinitialized by computing new initial probabilities $\vec{p}_i^{(0)}$. For units u_i which have been assigned names these are not actually probability vectors any more. The sum of their components adds up to k_i+1 where k_i is the number of assigned names. Since relations are used to compute initial

probabilities only when units are assigned (otherwise only features are used), this also has the advantage of improving the original estimates.

In the optimization problem ($\mathcal{P}$) the constraints are now:

$$\begin{cases} k_i \le \sum_{k=1}^{L_i} p_i(k) \le k_i+1 \\ p_i(\ell) = 1 \text{ for all names } \ell \text{ which have already been assigned to unit } u_i. \end{cases}$$

That is, we do not allow the confidence value for already assigned names to be changed from 1.

COMPARISON WITH OTHER APPROACHES AND EXPERIMENTAL RESULTS

We compared our approach with results from two other techniques: the relaxation procedure originally introduced by Rosenfeld, Hummel and Zucker [10] (algorithm RHZ) and the sequential matching procedure developed by Nevatia and Price [6] (algorithm NP). The nonlinear iterative updating formula of algorithm RHZ can be expressed as

$$p_i^{(n+1}(k) = \frac{p_i^{(n)}(k) Q_i(k)}{\sum_{\ell=1}^{L_i} p_i^{(n)}(\ell) Q_i(\ell)} \qquad (10)$$

where the Q_i's are computed in the same way as in Eq. (2). On this particular proble, algorithm RHZ has a tendency to converge toward ambiguous solutions. The reason is, of course that this algorithm does not take into account completely the coupling between units as reflected by the $c(i,k,j,\ell)$'s and fails to account for the notion of ambiguity. As shown in Eqs. (6) and (8) this is not the case with the algorithm described in this paper.

The basic comparison procedure used here is the same as used in algorithm NP, so that many of the results are identical. The major problems with the sequential method are the lack of a consistent method for the handling of multiple assignments and errors which are caused by the order in which the objects are selected for matching. For example, if a pair of identical objects are near each other, the sequential procedure can easily find the wrong assignment for the first object, thus forcing the second to also be incorrect, due to the limit of assigning one name to only one unit (units may be assigned several names). Since the relaxation procedure is considering pairs of objects, the correct assignments advance to the top, if there are relations connecting the two objects.

1. Results

We have applied this procedure to several different scenes or portions of scenes and will present results from two of these scenes. The first is a high altitude aerial image with a few major regions (a city and rural areas) and a number of linear features (a major highway, river channels, and roads along the channels), (see Fig. 2). The second is a closer view (of a different area), with 2 roads included in the model. The roads are described in sections because of the tendency of the linear feature extraction procedure to break long linear features into shorter segments, (see Fig. 3).

To illustrate how the relaxation procedure operates, Fig. 1 is a graph of the probabilities of various assignments for one object through a series of macro iterations. The values are given for each microiteration through a long sequence of macro iterations. Figures 2 and 3 show the final results, with the objects outlined and labeled. Many of the labels overlap since adjacent linear features are being presented.

REFERENCES

1. S. Rubin, "The ARGOS Image Understanding System," Ph.D. thesis, Computer Science Department, Carnegie-Mellon U., Pittsburgh, PA, 1978.

2. D. Smith, "Search Strategies for the ARGOS Image Understanding System," in Proc. Image Understanding Workshop, Los Angeles, Ca. Nov. 1979, pp. 42-46.

3. K. Price and R. Reddy, "Matching Segments of Images," IEEE Trans-PAMI Vol. 1, Jan, 1979, pp. 110-116.

4. R. Ohlander, K. Price and R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Comp. Graphics and Image Processing, Vol. 8, pp. 313-333, 1978.

5. R. Nevatia, and K.R. Babu, "Linear Feature Extraction and Description," to appear in Computer Graphics and Image Processing.

6. R. Nevatia and K. Price, "Locating Structures in Aerial Images," submitted for publication.

7. O.D. Faugeras and M. Berthod, Scene Labeling: An Optimization Approach," Proceeding of the IEEE Computer Society Conference on Pattern Recognition and Image Processing, pp. 318-396, Chicago, August 6-8, 1980.

8. O.D. Faugeras and M. Berthod, "Improving Consistency and Reducing Ambiguity in Stochastic Labeling: An Optimization Approach," submitted to the IEEE Trans. on Pattern Analysis and Machine Intelligence, November 1979.

9. M. Berthod and O.D. Faugeras, "Using Context in the Global Recognition of a Set of Objects: An Optimization Approach,"

10. A. Rosenfeld, R.A. Hummel and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. on Syst., Man, and Cybern. SMC-6, No. 6, pp. 420-453, June 1976.

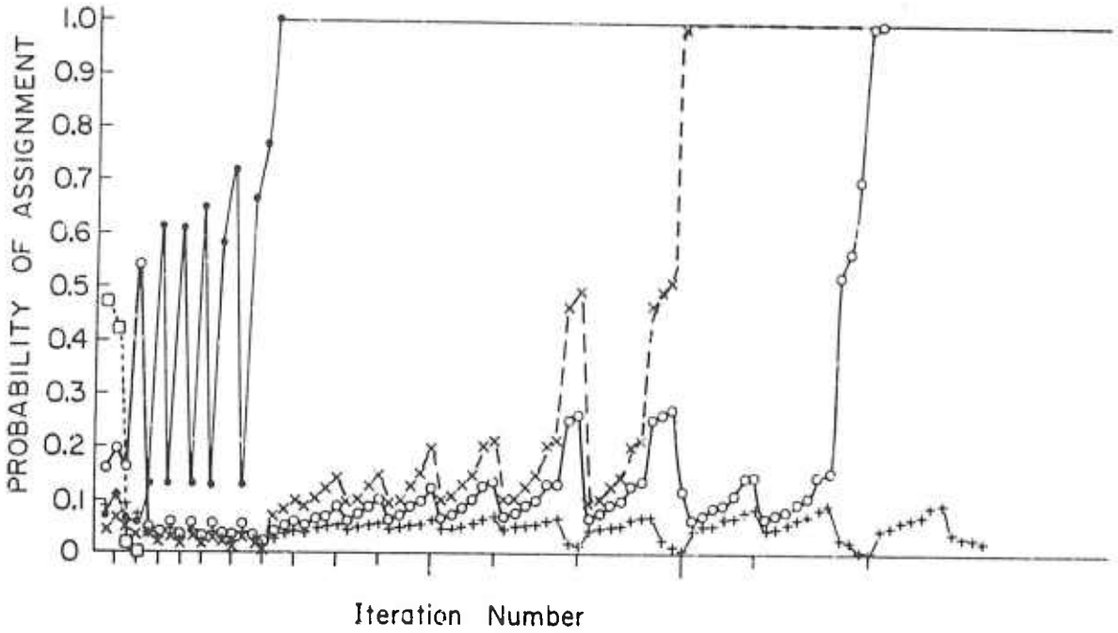


Fig. 1 Graph of the match likelihoods for several image segments to be assigned to one model element. The model element being considered is labeled "SOUTH-HIGHWAY" in the final results of Fig. 6. The tick marks along the horizontal axis indicate each Macro iteration. The three segments which rise to 1.0 are all correct assignments. Note also that the initial best assignments is not considered after the firm assignment made on the second macro iteration.

Fig. 2 Final assignments for Stockton area.

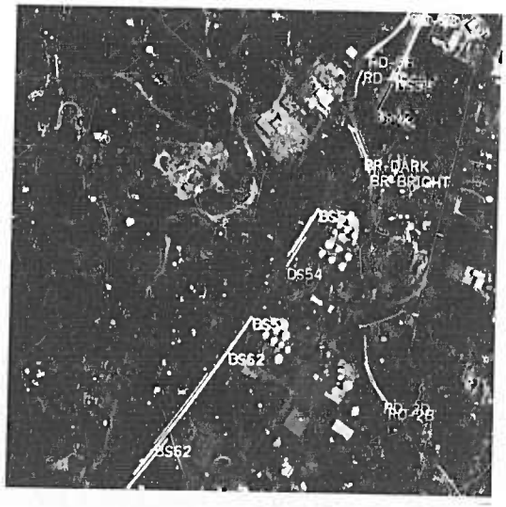


Fig. 3 Final assignments for road segments of Ft. Belvoir area.

REPRESENTING AND REASONING ABOUT PARTIALLY SPECIFIED SCENES

Rodney A. Brooks and Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

We present a representational scheme for specific and generic objects, partially specified scenes and partially specified camera models. It is built on top of our previous geometric volume based representations, and relies on specialization by constraints as its primitive element. We extend the specialization mechanism to enable case analysis in determining observational invariants of objects. A set of rules is given which aids in reasoning about geometric relationships between coordinate systems linked by multiple partially specified coordinate transforms. Together these two mechanisms provide powerful methods for predicting the appearances of objects. We demonstrate how to make use of two dimensional match information to interpret the image in a three dimensional model of the world. Again we make use of the specialization mechanism.

1. Introduction

The development of the ACRONYM model-based vision system has been previously described ([4], [5] and [6]). A brief overview of the system follows. This is intended to give a firm basis for the discussion of solutions to computational problems which arise when image interpretation is viewed as an interaction of prediction and description. This paper will be concerned primarily with prediction and use of predictions to interpret descriptions produced by low level image processing.

Figure 1 shows a block diagram of the major modules and data structures of ACRONYM. The data structures comprise the middle column. A user interacts with the system via a high level modeling language and an interactive editor to provide three dimensional descriptions of objects and object classes. A graphics module provides valuable feedback during this task. The models so constructed are volumetric descriptions based on generalized cones [1]. The models are tree structured and provide multiple levels of detail in their representation. All this has previously been described in detail (e.g. [6]). In section 2 we describe a new level of representation built on top of this.

A rule-based module, the predictor and planner, takes models of objects and scenes and produces the observability graph which is a prediction of the appearance of the objects expected within the scene, and a plan, or instructions, for descriptive processes and the matcher to find instances of the objects within the image. The process of prediction and planning is repeated as first coarse interpretations are found, more predictions are carried out, and finer, less ambiguous interpretations are produced. Section 3 describes some new mechanisms used for prediction, and the way they fit in with natural extensions to the additions to object representation described in section 2.

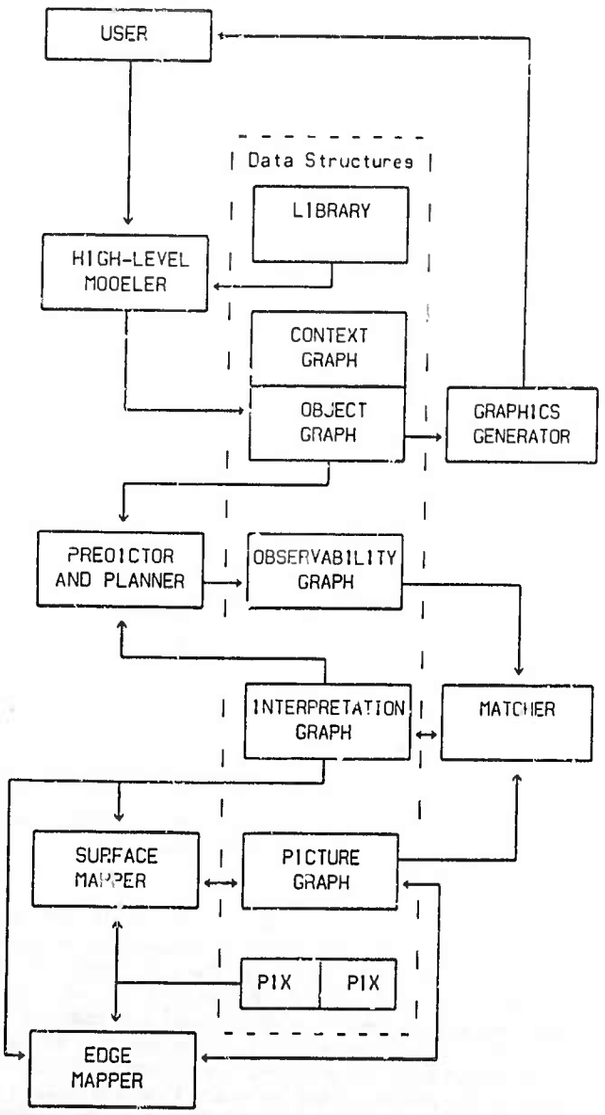


Fig. 1. The ACRONYM system.

Consider the lower portion of figure 1. The edge mapper describes pictures as ribbons and their spatial relationships. Ribbons are the two dimensional specialization of generalized cones [4]. Currently we deal only with monocular images, making use of the line finder of Nevatia and Babu [7] and the goal directed ribbon finder of Brooks [3]. Again, this edge mapping module may be invoked many times during the course of an interpretation as finer levels of interpretation are called for. Later ACRONYM will use stereo pairs of images and the surface mapper will extract depth information in the form of surface descriptions.

The matcher provides the interface between description and prediction. In all the image interpretations done by ACRONYM to date the matcher has been a syntactic graph matcher written by Russell Greiner [6]. It has been used to match the two dimensional predictions of the Observability graph to the two dimensional descriptions of the Picture graph. In section 3 we give the some details of the design of a new matcher to be implemented in the same rule set as that used for the predictor and planner. It will directly relate two dimensional descriptions back to three dimensional models, providing a strong mechanism for ensuring global consistency of local interpretations. It is expected that the matcher will gradually merge with the predictor and planner since they will interact with each other to a much greater degree in the new implementation.

2. Representation

We have previously ([4], [5], [6]) discussed ACRONYM's geometric representation of objects as volumes using generalized cones [1] as a volume primitive. We have alluded to, but not explained in detail, representation of generic classes and specific objects, explicit use of symmetry and representation of camera parameters. In this section we explain the aspects of our representation scheme devoted to these issues by means of an extended example.

The choice of representation scheme is driven both by (1) what we wish to represent, and (2) the class of computations we wish to base upon the representation.

(1) We need to represent generic classes of objects. This involves representing both variable sizes and variable structures with local constraints between such variations. We also wish to represent both subclasses and specific instances. Thus we will need to talk about both specialization and generalization. Our representation scheme is built on the specialization primitive. Thus generalization at the input level must be restructured to be a specialization at the representational level. We may incorporate generalization directly later.

(2) We need to reason about object classes and their specializations at the same time. This will allow matches to some part of an object to be carried down to a match to the corresponding part of a specialization of the object automatically. We need to be able to reason both about multiple instances of modeled objects, and multiple candidates for one or more instances, at the same time. Local interpretations provide more global constraints, but we need to know how global these constraints can safely be made. This must be explicitly represented in the model. For instance constraints on the length of a wing deduced from a local interpretation extend over both wings of a single aircraft, but they do not extend globally over all aircraft as there may be many types of aircraft in the image.

In the following discussion we will consider the problem of modeling the class of wide-bodied passenger jet aircraft, and specific wide-bodied passenger jet aircraft, such as the Boeing 747, Lockheed L-1011, McDonnell-Douglas DC-10 and the Airbus Consortium A-300. We will then model a wider situation where such aircraft are on runways and taxiways, and there are undetermined variables in the camera model.

Quantifiers

To represent the class of wide-bodied passenger jet aircraft we need to represent both variations in size (e.g. different aircraft subclasses will have different fuselage lengths), and variations in structure (e.g. different aircraft subclasses will have different engine configurations). In both cases we to represent the range of allowable variations. We refer to this as quantification of sets. Furthermore, there will sometimes be interdependencies between these variations (e.g. a scaling between fuselage length and wing span).

The primitive representational mechanism used in ACRONYM is that of units and slots. These are a slight generalization of atoms and properties of LISP. Objects are represented by units, as are generalized cones, cross-sections, sweeping-rules, spines, rotations and translations to name the more important ones. Figure 2 shows four units with their slots and fillers from a particular ACRONYM model. They describe the generalized cone representing the fuselage of the generic wide-bodied passenger jet aircraft. Note that units are referred to as "Nodes" because they are nodes of the Object graph of figure 1. The NAME slot is a distinguished slot which all units possess. It describes the entity represented by the unit and corresponds to the SELF slot of KRL units ([2]). Units identified by "Z" followed by a four digit number are those which were given no explicit identifier by the user who modelled the object. The modeling language parser has generated unique identifiers for them.

```
Node: FUSELAGE-CONE
  NAME:            SIMPLE-CONE
  SPINE:           Z0005
  SWEEPING-RULE:   CONSTANT-SWEEPING-RULE
  CROSS-SECTION:   Z0004

      Node: Z0005
        NAME:          SPINE
        TYPE:          STRAIGHT
        LENGTH:        FUSELAGE-LENGTH

      Node: CONSTANT-SWEEPING-RULE
        NAME:          SWEEPING-RULE
        TYPE:          CONSTANT

      Node: Z0004
        NAME:          CROSS-SECTION
        TYPE:          CIRCLE
        RADIUS:        FUSELAGE-RADIUS
```

Generalized cone representation of fuselage.
Fig. 2.

The value of a slot is given by its filler. Slot fillers may be immediate values, such as "2" or "STRAIGHT". They can also

be symbolic constants in the same sense as constants are used in programming languages such as PASCAL. Such fillers are fine for representing specific completely determined objects and situations. To allow for representation of variable quantities slots may be filled by a *quantifier*. A quantifier is a symbolic place holder for a slot filler. "FUSELAGE-LENGTH" and "FUSELAGE-RADIUS" are examples of such quantifiers in figure 2. Constraints can be placed on the values they are allowed to represent.

The following constraints might be imposed upon FUSELAGE-LENGTH and FUSELAGE-RADIUS when modeling the class of wide-bodied passenger jet aircraft:

```
FUSELAGE-LENGTH: (λ (V) (INTERVAL V 40.0 60.0))
FUSELAGE-RADIUS: (λ (V) (INTERVAL V 2.5 4.0))
```

These constrain the two quantifiers to be floating point quantities lying in the closed intervals [40.0, 60.0] and [2.5, 4.0] (the units are in meters). Thus the constraints say that a wide-bodied passenger jet aircraft has fuselage length ranging from 20 to 60 meters, and fuselage radius from 2.5 to 4 meters. In general a quantifier can have an arbitrary number of such constraints. We will discuss later the way in which these constraints are manipulated and used during prediction and interpretation.

Such quantifiers are thus used to express allowable variations in size of objects. They are also used to express variations in the allowable structure of objects. Figure 3 gives the complete subpart tree for a model of generic wide-bodied passenger jet aircraft. For brevity not all the slots of the OBJECT units are shown here. The QUANTIFIERS slot is explained in section 4. The SUBPARTS slot of an OBJECT unit is filled with a list of subparts giving the next level of description of the object. Entries in the list can be simple pointers to other OBJECT units (e.g. JET-AIRCRAFT has three substructures: STARBOARD-WING, PORT-WING and FUSELAGE). They can also be more complex such as the single entry for the subparts of STARBOARD-WING, which specifies a quantification of subparts called STARBOARD-ENGINE. The quantification description follows the same rules as unit slot fillers. In this case the quantification is the quantifier F-ENG-QUANT. Note that PORT-WING has a quantification of PORT-ENGINEs as subparts, which is represented by the same quantifier F-ENG-QUANT. Thus we have explicitly represented the symmetry of the aircraft: it has the same number of engines attached to each wing. Constraints on this quantifier and on R-ENG-QUANT, the number of rear engines might be:

```
F-ENG-QUANT: (λ (I) (INTERVAL I 1 2))
             (λ (I) (GREATERP 3 (+ I R-ENG-QUANT)))
R-ENG-QUANT: (λ (I) (INTERVAL I 0 1))
```

An alternate, but equivalent, second constraint for F-ENG-QUANT might be:

```
(λ (I) (MEMBER (LIST I R-ENG-QUANT)
               '((1 0) (1 1) (2 0))))
```

These say that there must be either one or two engines on each wing, zero or one at the rear of the aircraft, and if there are two on each wing then there are zero at the rear. We will return to this example in section 4.

Symmetry of size (such as length of the wings) can likewise be represented by using the same quantifier as a place holder in the appropriate pair of slots. Interdependencies on sizes, such as those on the number of forward and rear engines above, can be represented in the constraints on the quantifiers.

```
Node: JET-AIRCRAFT
  NAME:                OBJECT
  SUBPARTS:            (STARBOARD-WING PORT-WING
                        FUSELAGE)
  QUANTIFIERS:         (F-ENG-QUANT ENGINE-LENGTH
                        ENGINE-RADIUS
                        WING-ATTACHMENT ENG-OUT
                        ONE-WING-SPAN
                        WING-SWEEP-BACK
                        WING-LENGTH WING-RATIO
                        WING-WIDTH WING-THICK)

Node: STARBOARD-WING
  NAME:                OBJECT
  SUBPARTS:            ((SP-OES F-ENG-QUANT .
                        STARBOARD-ENGINE))
  CONE-DESCRIPTOR:     STARBOARD-WING-CONE

Node: STARBOARD-ENGINE
  NAME:                OBJECT
  CONE-DESCRIPTOR:     PORT-ENGINE-CONE

Node: PORT-WING
  NAME:                OBJECT
  SUBPARTS:            ((SP-OES F-ENG-QUANT .
                        PORT-ENGINE))
  CONE-DESCRIPTOR:     PORT-WING-CONE

Node: PORT-ENGINE
  NAME:                OBJECT
  CONE-DESCRIPTOR:     PORT-ENGINE-CONE

Node: FUSELAGE
  NAME:                OBJECT
  SUBPARTS:            (RUDDER STARBOARD-STABILIZER
                        PORT-STABILIZER)
  QUANTIFIERS:         (STAB-ATTACH STAB-WIDTH
                        STAB-THICK STAB-SPAN
                        STAB-SWEEP-BACK
                        STAB-RATIO)
  CONE-DESCRIPTOR:     FUSELAGE-CONE

Node: RUDDER
  NAME:                OBJECT
  SUBPARTS:            ((SP-OES R-ENG-QUANT .
                        REAR-ENGINE))
  CONE-DESCRIPTOR:     RUDDER-CONE

Node: REAR-ENGINE
  NAME:                OBJECT
  CONE-DESCRIPTOR:     REAR-ENGINE-CONE

Node: STARBOARD-STABILIZER
  NAME:                OBJECT
  CONE-DESCRIPTOR:     STARBOARD-STABILIZER-CONE

Node: PORT-STABILIZER
  NAME:                OBJECT
  CONE-DESCRIPTOR:     PORT-STABILIZER-CONE
```

Subpart tree of generic passenger jet.
Fig. 3.

Our complete model for a generic wide-bodied passenger jet aircraft has 28 quantifiers describing allowable variations in size and structure.

Restriction Nodes

It should be clear that to model a subclass of wide-bodied passenger jet aircraft we need only provide a different (more restrictive) set of constraints for the quantifiers used in the general model. To model a specific type of aircraft we could force the constraints to be completely specific (e.g. (λ (V) (= V 52.8))). Thus we will not need to distinguish between specialization of the general model to a subclass, or an individual. (Note that the notion of individual gets very confused here. Should a completely specified model of an SP-747, or 747-B, be considered an individual or a class, since there are multiple instances in the real world? By not distinguishing between individuals and classes at the model level we finesse the problem. If we were basing a natural language system upon our representation we might run into problems with this.)

Given that subclasses use different sets of constraints, the problem arises of how to represent multiple subclasses simultaneously. We introduce a new type of node to the representation: a restriction node. These are the embodiment of specialization. Restriction nodes form a tree, rooted at a distinguished node, the BASE-RESTRICTION. Constraints are always associated with a restriction node. A restriction node implicitly inherits all the constraints associated with its ancestors in the tree. Thus a daughter restriction node never has weaker constraints than its parents.

For the example of the generic wide-bodied passenger jet aircraft the constraints are associated with some restriction node, GENERIC-JET-AIRCRAFT say. Its parent would most likely be the BASE-RESTRICTION. To represent the class of 747s the following restriction node might be included:

```
Node: BOEING-747
  NAME:          RESTRICTION
  PARENT:        GENERIC-JET-AIRCRAFT
  TYPE:          MODEL-SPECIALIZATION
  CONSTRAINTS:   <list of constraints>
```

The CONSTRAINTS slot would be filled with the constraints additional to those in GENERIC-JET-AIRCRAFT necessary for representing the sub-class of Boeing 747s. This restriction might in turn have daughter restrictions to represent further subclasses such as SP-747s and 747-Bs.

Whenever a model is accessed (by the predictor and planner say), is is accessed in the context of a restriction node. Thus when reasoning about the generic class of wide-bodied aircraft the predictor and planner will access the JET-AIRCRAFT model and base its reasoning on the constraints given by the GENERIC-JET-AIRCRAFT restriction node. When reasoning about Boeing 747s it will base its reasoning about the JET-AIRCRAFT model on the constraints given by the BOEING-747 restriction node.

Variable Affixments

Affixments are coordinate transforms between local coordinate systems of objects. They are comprised of a rotation and a translation.

Sometimes affixments vary over an object class. For instance in the generic wide-bodied passenger jet aircraft the position along the fuselage at which the wings will be attached will vary with particular types of aircraft. Other times there may be an allowable variation in an affixment within a single object whenever an articulated object is modeled. Variable affixments will also be useful for modeling allowed spatial relationships between two objects - for instance an aircraft is on a runway.

Notationally we represent a vector as a triple (a,b,c) where a, b and c are scalars. We represent a rotation as a pair <v,m> where v is a vector, and m a scalar magnitude. An affixment will be written as a pair (r,t) where r is a rotation and t a translation vector. We will use some special vectors also: $\hat{x}$, $\hat{y}$ and $\hat{z}$. We use * for the composition of rotations, and ⊛ for the application of a rotation to a vector.

In ACRONYM itself, by treating vectors and rotations as units with slots we can use the quantifier mechanism to represent affixments which describe a class of coordinate transforms. (It turns out that in the ACRONYM implementation it is rather easy to extend the allowable fillers for rotations and translations to expressions involving quantifiers.) This gives symbolic representations for rotations and translations.

```
Node: RUNWAY
  NAME:              OBJECT
  CONE-DESCRIPTOR:   Z0025

      Node: Z0025
        NAME:            SIMPLE-CONE
        SPINE:           Z0026
        SWEEPING-RULE:   CONSTANT-SWEEPING-RULE
        CROSS-SECTION:   Z0027

            Node: Z0026
              NAME:      SPINE
              TYPE:      STRAIGHT
              LENGTH:    RUNWAY-LENGTH

            Node: CONSTANT-SWEEPING-RULE
              NAME:      SWEEPING-RULE
              TYPE:      CONSTANT

            Node: Z0027
              NAME:      CROSS-SECTION
              TYPE:      RECTANGLE
              WIDTH:     RUNWAY-WIDTH
              HEIGHT:    0.0
```

Model of runway
Fig. 4.

Now consider the problem of representing the fact that an aircraft is somewhere on a runway. Suppose the runway is represented as in figure 4. In this case its coordinate system will have its x axis centered along the length of the runway, the y axis perpendicular at one end, and the positive z direction will be vertically straight up. Suppose that the coordinate system for the aircraft has its x axis running along the spine of the

fuselage, and has its z axis skyward for the standard orientation of an airplane. Thus to represent the aircraft being on the runway we could affix it with the following affixment:

(<$\hat{z}$, ORI), (JET-RUNWAY-X, JET-RUNWAY-Y, 0))

where ORI, JET-RUNWAY-X and JET-RUNWAY-Y are quantifiers with the following constraints:

```
JET-RUNWAY-X:   (λ (V)
                   (INTERVAL V 0.0 RUNWAY-LENGTH))
JET-RUNWAY-Y:   (λ (V)
                   (INTERVAL
                    V
                    (TIMES RUNWAY-WIDTH -0.5)
                    (TIMES RUNWAY-WIDTH 0.5)))
```

These constrain the aircraft to be on the runway, in the normal orientation for an airplane (i.e. not upside down or any such), but it does not constrain the direction in which the aircraft is pointed. If we wished to constrain the aircraft to approximately line up in the direction of the runway we could include a constraint on the quantifier ORI, allowing for some small error.

Partially Specified Camera Model

If we are to predict the appearance of objects it will help to have some model of their position and orientation relative to the camera. We have chosen to use a world coordinate system into which both objects and camera are placed via affixment type coordinate transforms. This allows the user modelling a scene to think in familiar terms. The affixments so used are of course allowed to include quantifiers in their specification.

The camera has a local coordinate system where focal point is centered at the origin, the viewing direction is along the negative z axis, and the top of the image screen is in the positive y direction.

Suppose we are modeling the situation where the camera is onboard an aircraft flying over some scene and the camera is pointed directly downwards. Suppose further that the exact height of the aircraft is unknown. If the objects on the ground are modeled with unknown ground coordinates (there may well be mutual constraints between the quantifiers representing the positions of the objects on the ground) then we can choose a world coordinate system so that the camera is directly over the origin, with its x and y axes directly over the world x and y axes. Then letting HEIGHT be the quantifier representing the height of the camera above ground we can affix the camera to the world with:

(i, (0, 0, HEIGHT))

where i is the identity rotation. The quantifier HEIGHT might be constrained according to some a priori knowledge of the conditions under which the photographs to be analyzed were taken. If we want to include the effects of roll and pitch of the aircraft carrying the camera (yaw has already been taken care of by our choice of world coordinates) we could apply two rotations to the above affixment, obtaining the expression:

(<$\hat{x}$, PITCH>*<$\hat{y}$, ROLL>, (0, 0, HEIGHT))

Again, the quantifiers PITCH and ROLL will have constraints generated from knowledge of the conditions at the time the photographs were taken.

When the predictor and planner needs to deduce the position of an object relative to the camera coordinates it will need to invert the above coordinate transform. This can be achieved by a simple set of manipulation rules, and in this case the result would be:

(<$\hat{y}$, -ROLL>*<$\hat{x}$, -PITCH>,
<$\hat{y}$, -ROLL>*<$\hat{x}$, -PITCH>•(0, 0, -HEIGHT))

3. Geometric Reasoning and Case Analysis

Invariant observables are image features which will be observable over all possible viewing conditions (e.g. collinearities, connectivity stemming from three dimensional connectivity). Quasi-invariants are those that will be observable over the possible range of variation within the modeled scene (e.g. the fuselage of an aircraft on the ground will appear as a straight ribbon from aerial images taken at any altitude).

The predictor and planner needs to be able to detect when quasi-invariants are available. Thus it must be able to reason about the spatial relationship between the camera and a model. This relationship will usually be only partially specified, and will involve a number of quantifiers. Often quasi-invariants will not be directly available. Instead the predictor and planner will need to break up the possible spatial relationships into sub-cases where there are quasi-invariants. Again it will need to reason about a chain of partially specified coordinate transforms. In addition it will need to produce observability predictions for the different sub-cases, where the constraints on quantifiers have been specialized further than in the modeled situation.

Reasoning About Coordinate Transforms

The orientation and position of an object relative to the camera will be given by a series of affixments, which are pairs of rotations and translations. The final orientation will be given by the product of the rotational components of these affixments. The relative orientation is important for making predictions of the appearance of the object. The position is given by a sum of applications of rotation expressions to individual translation vectors. The positions of objects are important for determining whether they are visible and for the predicting spatial relationships between objects in the image.

In this section we will only deal with methods for manipulating and understanding products of rotations. These methods are symbolic in nature and allow the predictor and planner to reason about rotations which are expressed in term of quantifiers. We are working on methods for dealing with the results of applying rotations to vectors; the other half of understanding compositions of affixments. We have developed some rules for this, but will report on them at a later date after further work.

Rotations of three space form a group under the operation of composition. They are associative but not commutative. Commutivity would help greatly in simplifying products of rotations as it is easy to compose rotations which share their axis, into a single rotation. There is a slightly weaker

property of three dimensional rotations however. Its proof is straightforward but tedious, and is omitted here. Let v_1 and v_2 be two three-space vectors and m_1 and m_2 be two scalars. Then the following two identities are true:

$$\langle v_1, m_1\rangle * \langle v_2, m_2\rangle = \langle v_2, m_2\rangle * \langle (\langle v_2, -m_2\rangle \oplus v_1), m_1\rangle$$

$$\langle v_1, m_1\rangle * \langle v_2, m_2\rangle = \langle (\langle v_1, m_1\rangle \oplus v_2), m_2\rangle * \langle v_1, m_1\rangle$$

These will allow us to "shift" rotations both to the left and to the right. The only problem is that as a rotation is shifted it leaves rotations with complex axis expressions in its wake. There is a subgroup of rotations for which these axis expressions are no more complex than the original. This is the group of 24 rotations which permute the x, y and z axis amongst themselves and their negations. When they are used with the above identities the new axis expression is a permutation of the original axis, with perhaps some sign changes.

We will be particularly interested in a subset of that rotational subgroup. It consists of the identity rotation i, and rotations about the three coordinate axes whose magnitudes are multiples of $\pi/2$. We write them x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2 and z_3. The subscript indicates the magnitude of the rotation as a multiple of $\pi/2$. We will call these ten rotations elementary. It turns out that they are very commonly used rotations in modeling man-made scenes. Rotation expressions for the orientation of an object will often be chiefly composed of elementary rotations.

Elementary rotations are closed under the identities given above. For example:

$$x_3 * y_1 = y_1 * z_3$$
$$x_3 * y_1 = z_3 * x_3$$

Using the general identities given above we can simplify expressions of products of rotations which include elementary rotations by "moving" them, and multiplying out adjacent elementary rotations which share the same axis. In particular, using the following five simplification rules we can remove all but at most two elementary rotations from a product of rotations. There will be at most one elementary rotation at the left of the expression, which will be one of z_1, z_2 or z_3. At the right of the expression there may be one of x_1, x_2, x_3, y_1 or y_3. The other rotations will merely have had the components of their axis of rotation permuted and perhaps negated.

SR1: Compose adjacent elementary rotations sharing the same axis of rotation.

SR2: Move instances of z_1, z_2 and z_3 to the left applying **SR1** whenever possible.

SR3: Move the left-most x-axis elementary rotation to the right until it is adjacent to another, or it is the right most rotation in the expression. Move each z-axis elementary rotation which may have been introduced to the left of the expression. Apply **SR1** to the result wherever necessary.

SR4: Move elementary y-axis rotations to the right, stopping before any elementary x-axis rotation which might be there. Apply **SR1** whenever necessary.

SR5: Make substitutions at the right of the expression using the following identities and shift any introduced elementary z-axis rotations to the left of the expression, applying SR1 wherever necessary.

$$y_1 * x_1 = z_3 * y_1,\ y_1 * x_2 = z_2 * y_1,\ y_1 * x_3 = z_1 * y_1$$
$$y_3 * x_1 = z_1 * y_3,\ y_3 * x_2 = z_2 * y_3,\ y_1 * x_3 = z_3 * y_3$$
$$y_2 = z_2 * x_2$$

As an example (albeit more complex than is usually found in aerial images - the following orientation expression comes from a ground level camera with small TILT and PAN, viewing a subpart of a complex object which is oriented upright, but arbitrarily on the ground) consider the following rotation expression:

$\langle \hat{x}, -TILT\rangle * \langle \hat{y}, PAN\rangle * z_3 * y_3 * \langle \hat{z}, ORI\rangle * y_3 * y_1 * y_1$

When the above rules are applied it simplifies to:

$z_3 * \langle \hat{y}, -TILT\rangle * \langle \hat{x}, -PAN\rangle * \langle \hat{z}, -ORI\rangle$

The reason for wanting the particular form for the rotation expression that the above rules supply is twofold. A left-most rotation about the z axis corresponds to a rotation of the image plane. Thus it can be ignored for the purpose of predicting appearances. The possibilities for the right-most elementary rotation (in the above case it is i, the identity) correspond to the six views of a generalized cone from along each axis ray in its local coordinate system. Many cones will be symmetric with respect to the x-axis rotations, as it corresponds to rotation about the spine, and y_1 and y_3 correspond to viewing the cross section at each end. Of course the non-elementary rotations remaining may cause problems. If the above expression is being used to predict the appearance of a cylinder, then all the right most rotations about the x axis can be ignored. Cylinders are invariant with respect to rotations about their spine. Thus the above expression could be treated equivalently to:

$\langle \hat{y}, -TILT\rangle$

For highly constrained TILT, this too could be ignored. We will not always be so lucky, but there will often be substantial simplifications.

Consider as a second example the camera orientation developed at the end of section 2 for the camera with pitch and roll and included. Suppose it is viewing an airplane sitting on the ground with some arbitrary orientation. Let ORI be the quantifier which corresponds to that degree of freedom. Then the orientation expression for the rudder will be:

$\langle \hat{y}, -ROLL\rangle * \langle \hat{x}, -PITCH\rangle * \langle \hat{z}, ORI\rangle * y_3$

The five simplification rules above leave this expression invariant. Using the first of the identities given at the beginning of the section the predictor and planner can symbolically shift the unconstrained rotation to the left, so that the expression becomes:

$$\langle \hat{z}, ORI\rangle * \langle (\langle \hat{z}, -ORI\rangle \oplus \hat{y}), -ROLL\rangle * \langle (\langle \hat{z}, -ORI\rangle \oplus \hat{x}), -PITCH\rangle * y_3$$

The left most rotation can now be ignored as it corresponds only to a rotation of the image plane. If ROLL and PITCH are constrained to be reasonable then the appearance of the airplane will be quasi-invariant, as effects from the two remaining rotations will be dependent on the cosines of ROLL and PITCH which will be close to one. Thus the view of the rudder will be that given by rotating it by y_3 - the view from its spine looking back at its top cross section.

Case Analysis

For a given object model with a given range of possible sizes, structures and orientations, it may not be possible to find adequate quasi-invariants to predict the appearance of the object. However by splitting up the range of allowed values for some quantifier, it will often be possible for the predictor and planner to produce subcases where quasi-invariants do exist.

Consider for example images of oil tanks taken from a downward looking camera from a relatively low altitude aircraft. Depending on their lateral distance from the aircraft the tanks will appear as circles, or as ellipses with short parallel sided ribbons connected to them. Rules to make such predictions are included in the predictor and planner. Further, both predictions tan be made if they are predicated on the value of the lateral distance. To to this new restriction nodes are introduced. They provide constraints on the appiopriate quantifiers, breaking the possiblities into cases. Predictions are attached to the appropriate restriction node. The following restriction nodes are typical:

```
Node: Z0137
  NAME:         RESTRICTION
  PARENT:       GENERIC-OIL-TANK
  TYPE:         OBSERVABILITY-CASE-ANALYSIS
  CONSTRAINTS:  (POS-X: (λ (V)
                           (> (+$ (*$ V V)
                                  (*$ POS-Y POS-Y))
                              1.0E6))
                POS-Y: (λ (V)
                           (> (+$ (*$ V V)
                                  (*$ POS-X POS-X))
                              1.0E6)))

Node: Z0138
  NAME:         RESTRICTION
  PARENT:       GENERIC-OIL-TANK
  TYPE:         OBSERVABILITY-CASE-ANALYSIS
  CONSTRAINTS:  (POS-X: (λ (V)
                           (< (+$ (*$ V V)
                                  (*$ POS-Y POS-Y))
                              1.44E6))
                POS-Y: (λ (V)
                           (< (+$ (*$ V V)
                                  (*$ POS-X POS-X))
                              1.44E6)))
```

The first node is for the case of the oil tank being greater than 1000 meters from the center of the point of view. The second for less than 1200 meters. It is quite alright that these restriction nodes are not mutually exclusive, as they will be used in a backward mode during matching, rather than a forward mode. That is to say, matches for the oil tank model will be hypothesized from some data, according to one of the two observability descriptions attached to the above constraint nodes. The appropriate constraints will be assumed on the basis of which match is hypothesized. By not allowing hypothesis of matches which lead to inconsistent constraints the implications of local matches are propagated to enforce global consistency.

4. From 2-D back to 3-D

Previously ([4], [5], [6]) we have described the general structure of the observability graph. We will not repeat that here. Instead we give a detailed example of how two dimensional matching can be used to understand a three dimensional scene. We also discuss some of the mechanisms for combining the local results of such matchings to produce a global understanding of the scene.

Predicting Uncertain Size

Consider the problem of predicting the length of the ribbon which will correspond to the fuselage of an aircraft as modeled by the generalized cone of figure 2. Suppose that the image is an aerial view taken by a camera at a height represented by the quantifier HEIGHT.

A simple approach (and that previously used in ACRONYM) is to calculate the extreme values allowable for the FUSELAGE-LENGTH and HEIGHT and hence calculate an upper and lower bound on the possible length of the ribbon in the image. Then at match time ribbons whose length falls within this range (and similarly for the width and taper) are accepted as candidate fuselage matches. Later constraints about spatial relations with other ribbons are used to confirm or reject the match.

This is unsatisfying however as such local matches provide no clue to the three dimensional size of the object, nor the height of the camera. A match where the ribbon is at the smaller end of the available range implies that HEIGHT is really constrained to its larger values. Thus a match for a wing which lies near the upper end of possible lengths for that ribbon would be inconsistent as it would imply that HEIGHT really takes on one of its lower possible values.

We therefore associate a restriction node with each potential match. The TYPE slot of the node distinguishes it from other types of restrictions (HYPOTHESIS-MATCH is used). The parent restriction node is that associated with the observability node being matched. The match itself provides new constraints on some quantifiers. If these constraints are inconsistent with those provided by the parent restriction then the match is immediately rejected. Otherwise the constraints are attached to the new restriction node. Later, as various local matches are combined these constraints are checked for consistency to see whether the local matches are globally consistent.

We now work through the details of the fuselage example. For a projective imaging system the observed distance m between two points distance l apart on the ground is given by:

$$m = \frac{cl}{h}$$

where c is a constant dependent on the focal distance of the camera and h is the height of the camera above ground. We will use the symbols l for FUSELAGE-LENGTH and h for HEIGHT for brevity in the following equations. The

constraints from the restriction node associated with the observability node may implicitly provide upper and lower bounds on these quantifiers. The predictor and planner has rules which examine sets of constraints and try (heuristically) to deduce such bounds. Suppose the lower bounds are l_i and h_i and the upper bounds are l_s and h_s. Then the simple acceptance condition on m is:

$$\frac{l_i c}{h_s} \leq m \leq \frac{l_s c}{h_i} .$$

However we can interpret the above equation a different way. If m is the observed or measured length of a ribbon, and it does indeed correspond to a fuselage of the class of aircraft we are searching for, then the following must be true of the length l of the fuselage, and the height h of the camera:

$$\frac{h_i m}{c} \leq l \leq \frac{h_s m}{c}$$

and:

$$\frac{l_i c}{m} \leq h \leq \frac{l_s c}{m}$$

Thus if we know bounds on one of l or h, then we are able to calculate bounds for the other which would be implied by accepting a match to the prediction. If these new bounds are inconsistent with those already known then the match should be rejected.

```
FUSELAGE-LENGTH:
        (λ (V)
           (INTERVAL V
                     (*$ (LOWERB HEIGHT)
                         0.0015)
                     (*$ (UPPERB HEIGHT)
                         0.0015)))

HEIGHT:
        (λ (V)
           (INTERVAL V
                     (//$ (LOWERB FUSELAGE-LENGTH)
                          0.0015)
                     (//$ (UPPERB FUSELAGE-LENGTH)
                          0.0015)))
```

Constraints generated by a match.
Fig. 5.

Figure 5 gives examples of the sort of constraints that might be generated at match time for c equal to 20 and a measured length of the ribbon m of 0.03. Note that the observability prediction itself would contain some code to generate this constraint rather than the constraint itself as m will not be known at prediction time. The functional forms UPPERB and LOWERB are interpreted by the constraint consistency checking mechanism to mean that upper and lower bounds for the named quantifier should be deduced from the other constraints known if possible. Notice too that the constraints should remain true when further information is learned about l or h. By re-evaluating these constraints at appropriate times we will be able to determine at a later time whether this local match is consistent with a more global interpretation. This is a great strength of the system we are describing.

From our experience with carrying out actual interpretations we find that any predictions should allow generously for errors (often 40-50%). This is because of the high error rate in the descriptive processes. Such a policy means that besides the desired matches, many new incorrect matches are made at the local level. However because of the structured nature of the models and the resulting inter-constrained observability predictions any such incorrect matches are rejected at later stages due to global inconsistencies. We have not encountered a single case of an incorrect match surviving to the final interpretation. The converse is not true. There have been a number of cases where local matches have been missed, even with large error margins allowed. More powerful descriptive methods would help this problem.

From Local to Global

After each phase of local matching, the matcher combines them into more global interpretations. This involves finding consistent subgraphs of matches. Previously consistency has only concerned the existence of arcs describing relations between matched ribbons. With the introduction of constraints on quantifiers during the ribbon matching process, these too must be checked for consistency.

Constraints on a quantifier at different HYPOTHESIS-MATCH restriction nodes may actually refer to different quantities in the scene. For instance each potential match for an aircraft may have constraints on FUSELAGE-LENGTH and on HEIGHT. When combining the matches for aircraft to produce an interpretation of the image, there is no reason to require that the constraints on FUSELAGE-LENGTH at these different nodes be mutually consistent. Different instances of wide-bodied passenger jet aircraft will be different lengths. However all the constraints on HEIGHT should be mutually consistent, as there is only one HEIGHT of the camera.

Sometimes when constraints on quantifiers actually correspond to different quantities in the world, it may be that these quantities should have the same value. For instance the ENGINE-LENGTH for the port and starboard engines correspond to physical measurements of different objects in the world. However since aircraft are symmetric the constraints given by the matches on possible values of ENGINE-LENGTH for each engine should be consistent. Thus when clumping the local matches for an aircraft the ENGINE-LENGTH constraints from each submatch should be checked for consistency. If they are not consistent the particular set of local matches should be rejected as inconsistent.

A slot is provided in object units to represent which quantifiers matched at a lower level should be held consistent for interpretation of an object. This is the QUANTIFIERS slot as shown in figure 3. As the matcher is combining local matches it looks up the subpart tree. Any quantifier mentioned in a QUANTIFIERS slot of any ancestor of the object has its constraints copied into the restriction node for the new more global node. As each constraint is introduced it is checked for consistency. This process is not quite straight forward as sometimes a constraint on a quantifier is in terms of another quantifier which is not being brought into the new match. Such

is the case of FUSELAGE-LENGTH and HEIGHT from figure 5, when a global interpretation is being made involving many aircraft. The constraints on HEIGHT must be replaced with ones which use the current lower and upper bounds determinable for FUSELAGE-LENGTH for each aircraft.

5. Conclusion

We have given details of our representational scheme for generic and specific objects and incompletely specified scenes. We have given explicit rules to help with geometric reasoning, and methods for representing multiple predictions about the same object. We have shown how to use two dimensional match information to infer three dimensional information. All these rely heavily on the representational tools of quantifiers and restriction nodes. We have described three uses for restriction nodes:

(a) producing models which are instances or subclasses of class models.
(b) providing a mechanism for case analysis in prediction.
(c) allowing multiple matches against single models and subsequent independent reasoning about each instance.

The three types of restriction node are tagged and hence distinguishable when necessary. However the three cases involve many similar computational problems, and the common representation allows the reasoning system to handle these common computational problems with precisely the same mechanisms.

We have demonstrated how constraints from individual local matches can be combined to produce globally consistent interpretations.

References

[1] Binford, Thomas O., "Visual Perception by Computer," Invited paper at *IEEE Systems Science and Cybernetics Conference*, Miami, Dec. 1971.

[2] Bobrow, Daniel G. and Terry Winograd, "An Overview of KRL, a Knowledge Representation Language," *Cognitive Science* 1, 1977, 3-46.

[3] Brooks, Rodney A., "Goal-Directed Edge Linking and Ribbon Finding," *Proc. ARPA Image Understanding Workshop*, Palo Alto, Apr. 1979, 72-78.

[4] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "A Model-Based Vision System," *Proc. ARPA Image Understanding Workshop*, Cambridge, May 1978, 36-44.

[5] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "Progress Report on a Model-Based Vision System," *Proc. ARPA Image Understanding Workshop*, Pittsburgh, Nov. 1978, 145-151.

[6] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "The ACRONYM Model-Based Vision System," *Proc. of IJCAI-79*, Tokyo, Aug. 1979, 105-113.

[7] Nevatia, Ramakant and K. Ramesh Babu, "Linear Feature Extraction and Description," *Proc. of IJCAI-79*, Tokyo, Aug. 1979, 639-641.

A STORAGE REPRESENTATION FOR EFFICIENT ACCESS TO LARGE MULTI-DIMENSIONAL ARRAYS

Lynn H. Quam
SRI International
Menlo Park, California 94025

ABSTRACT:

This paper addresses problems associated with the access of elements of large multi-dimensional arrays when the order of access is either unpredictable or is orthogonal to the conventional order of array storage. Large arrays are defined as arrays which are larger than the physical memory immediately available to store them. Such arrays must be accessed either by the virtual memory system of the computer and operating system, or by direct input and output of blocks of the array to a file system. In either case, the direct result of an inappropriate order of reference to the elements of the array is the very time-consuming movement of data between levels in the memory hierarchy, often costing factors of three orders of magnitude in algorithm performance.

The access to elements of large arrays is decomposed into three steps: the transformation of the subscript values of an n-dimensional array into the element number in a 1-dimensional virtual array, the mapping of virtual array position to physical memory position, and the access to the array element in physical memory. The virtual to physical mapping step is unnecessary on computer systems with sufficiently large virtual address spaces. This paper is primarily concerned with the first of these steps.

A subscript transformation is proposed which solves many of the order-of-access problems associated with conventional array storage. This transformation is a based on an additive decomposition of the calculation of element number in the array into the sum of a set of integer functions applied to the set of subscripts as follows:

element-number(i,j,...) = fi(i) + fj(j) + ...

The choices for the transformation functions which minimize access time to the elements of the array depend on the characteristics of the memory hierarchy of the computer system and the order of accesses to the elements of the array. It is conjectured that given appropriate models for system and algorithm access characteristics, then a pragmatically optimum choice can be made for the subscript transformation functions. In general, these models must be stochastic, but in certain cases deterministic models are possible.

The use of tables to evaluate the functions fi and fj make the implementation very efficient using conventional computers. When the array accesses are made in an order inappropriate to conventional array storage order, this scheme requires far less time than for conventional array accessesing schemes, otherwise the accessing times are comparable.

The semantics of a set procedures for array access, array creation, and the association of arrays with file names is defined. For computer systems with insufficient virtual memory, such as the PDP-10, a software virtual to physical mapping scheme is given in appendix III. Implementations are also given there for the VAX and PDP-10 series computers are to access pixels of large images stored as 2-dimensional arrays of n bits per element.

INTRODUCTION:

This paper addresses problems associated with the access of elements of large multi-dimensional arrays when the order of access is either unpredictable or is orthogonal to the conventional order of array storage. Large arrays are defined as arrays which are larger than the physical memory immediately available to store them. Such arrays must be accessed either by the virtual memory system of the computer and operating system, or by direct input and output of blocks of the array to a file system. In either case, the direct result of an inappropriate order of reference to the elements of the array is the very time-consuming movement of data between levels in the memory hierarchy, often costing factors of three orders of magnitude in algorithm performance.

The access to elements of large arrays is decomposed into three steps: the transformation of the subscript values of an n-dimensional array into the element number in a 1-dimensional virtual array, the mapping of virtual array position to physical memory position, and the access to the array element in physical memory. The virtual to physical mapping step is unnecessary on computer systems with sufficiently large virtual address spaces. This paper is primarily concerned with the first of these steps.

The subscript transformation which is conventionally used is a linear combination of the subscript values, of the form:

element-number(i,j,k, ...) = a + b*i + c*j + ...

A more general subscript transformation is proposed which is believed to solve most of the order-of-access problems associated with conventional array storage. This transformation is a based on an additive decomposition of the calculation of element number in the array into the sum of a set of integer functions applied to the set of subscripts as follows:

element-number(i,j,k,...) = fi(i) + fj(j) + ...

A CASE STUDY The Storage of Large Images

The storage of large images is a special case of the storage of large arrays. It has been common practice in digital image processing and image analysis to represent images as two dimensional arrays of pixels stored in conventional order, for example, by rows in a left to right, top to bottom manner referred to as "row order". Algorithms that access the pixels in the same order that they were stored can efficiently handle even very large images in one pass through an image stored on secondary storage.

There are, however, many image analysis algorithms which do not access the pixels in strict row order. Geometric transformations such as image rotation and transposition have a very predictable order of access, but it can be totally orthogonal to the storage order. Such algorithms have deterministic models for accessing order.

As an example, consider a straight-forward implementation of array transposition as B[i,j]=A[j,i] applied to a 2048x2048 image which is conventionally stored one pixel per byte. Assume the system uses any page replacement algorithm related to the least-recently-used (LRU) algorithm. Under those conditions, this array transposition calculation would cost a page fault on every pixel if the entire image would not fit into physical memory. On a VAX computer system running either VMS or Berkeley UNIX, with a page size of 1K bytes, and a cost of approximately 30 milliseconds per page fault, this taak would consume 18 HOURS, only one minute of which is actual proceasing. Other systems such as KL-10 computers running TOPS-20 produce comparable results.

The order of access of algorithms such as edge following and region growing ia not deterministic, since that order ia data dependent, but it can be modelled as a atochastic process. There are mechanisms for implementing many of these algorithma such that they access their data in storage order, but usually these mechanisms are far more complex than the image analysis algorithms themselves and introduce other inefficiencies or aevere restrictions.

Reddy (1) and others have propoaed that large images ba stored not by rows, but by rectangular blocks which ara roughly equal to the size of a virtual memory page. With such a repreaentation, any order of accesa to the image with a reasonable degree of 2-d locale of reference will involve a only small number of pages at a time. For example, assume we have a 2048x2048-bit image with a block size of 32x32 pixels (1K pixels per block), and a page size of 1K bytes. To get the pixels from any row or column requires access to 64 pages. For an image stored in row order, the rows require access to 2 pages, and the columns require access to 2048 pages. To extract a 128x128 window from an image stored by blocks requires access to a minimum of 16 pages and a maximum of 25 pages depending on the alignment of the window with the 32x32 block boundaries. Similarly, to extract the same window from an image stored in row order requires access to a range of from 128 to 256 pages.

The major issues which must be addressed to achieve widespread acceptance of such a block oriented image representation are the efficiency of access to the pixels. the flexibility in the arrangement of the pixels, and the ease and transparency of use in a high level language.

PROPOSED SUBSCRIPT TRANSFORMATION:

The proposed transformation from array subscript values to the storage position in a 1-dimensional virtual array is of the form:

element-number(i,j,k, ...) = fi(i) + fj(j) + ...

The following examples illustrate some of the different choices of the transformation functions for 2-dimensional arrays:

Example 1: Storage by row, ni elements per row:

element-number(i,j) = i*ni + j

or
fi(i) = i*ni
fj(j) = j

Example 2: Storage by column, nj elements per per column:

element-number(i,j) = i + j*nj

or
fi(i) = i
fj(j) = j*nj

Example 3: Storage by block:

Given an array of ni rows and nj columns, it ia often uaeful to divide the array into rectangular blocka which contain bi rows and bj columna. Usually, bi*bj will be chosen to be related to the virtual memory page size, probably in the range of 512 to 2048 bytea, but it is not necessary to require alignment of array blocks with virtual memory pages. (In the representation proposed here, array rows and columns must be padded to exactly fill an integral number of array blocks to enable the additive decomposition of the transformation.)

Aasuming that elements are stored in blocks in row order, and that blocks are arranged in row order, then follcwing calculations compute the element number within array:

```
element-number(i,j) =
                  (j mod bj)
                + (i mod bi) * bj
                + (j div bj) * bi*bj
                + (i div bi) * bi*bj * (nj div bj)
or      fi(i) =   (i mod bi) * bj
                + (i div bi) * bi*bj * (nj div bj)

        fj(j) =   (j mod bj)
                + (j div bj) * bi*bj
```

In the proposed accessing scheme, the values of fi and fj are pre-calculated over the ranges of each subscript and stored in tables, rather than calculated on every access to the array.

Example 4: Storage by nested blocks:

A generalization of the block storage scheme is the division of the array into a nested sequence of blocks. At each level k in the sequence the block consists of bi[k] x bj[k] sub-blocks of level k-1. Block level 0 consists of a single array element.

Assume that the array has been padded to dimensions ni and nj which have as factors the desired block dimensions:

$$ni = bi[1] * bi[2] * \ldots * bi[nk]$$
$$nj = bj[1] * bj[2] * \ldots * bj[nk]$$

define $pi[0] = 1$, $pi[k] = pi[k-1] * bi[k]$
and $pj[0] = 1$, $pj[k] = pj[k-1] * bj[k]$

then fi(i) =

$$\sum_{k=1,nk} ((i \text{ div } pi[k-1]) \bmod bi[k]) * pi[k-1] * pj[k]$$

and fj(j) =

$$\sum_{k=1,nk} ((j \text{ div } pj[k-1]) \bmod bj[k]) * pi[k-1] * pj[k-1]$$

An interesting case of the nested block representation is the binary case, where for all k, bi[k]=bj[k]=2. This subscript to element-number transformation would cause a 2-d array to be stored as follows:

```
 0  1  4  5 16 17 20 21  64  65  68  69 ...
 2  3  6  7 18 19 22 23  66  67  70  71 ...
 8  9 12 13 24 25 28 29  72  73  76  77 ...
10 11 14 15 26 27 30 31  74  75  78  79 ...
32 33 36 37 48 49 52 53  96  97 100 101 ...
34 35 38 39 50 51 54 55  98  99 102 103 ...
40 41 44 45 56 57 60 61 104 105 108 109 ...
42 43 46 47 58 59 62 63 106 107 110 111 ...
 .  .  .  .  .  .  .  .   .   .   .   .
 .  .  .  .  .  .  .  .   .   .   .   .
 .  .  .  .  .  .  .  .   .   .   .   .
```

The primary advantage of the binary nested block representation is that it is directionally isotropic in efficiency of access at all scales, from very small array element neighborhoods to large blocks of very large arrays. This arrangement is useful because of the hierarchical structure of computer memory.

ORDER OF ACCESS AND MEMORY HIERARCHY:

Nearly all computer systems have at least two levels of memory hierarchy: the main memory of the CPU and a disk file system. Many newer computers have an additional level of memory hierarchy which consists of a high speed cache between the CPU and the main memory. It the near future high speed page caches build from bubble memory elements are expected. Most of these levels of memory exist because of tradeoffs between cost per bit and access time.

It is conjectured that given a model for the characteristics of the memory hierarchy together with a model for the order of accesses made to the elements of an array stored in that memory hierarchy, then a pragmatically optimum choice can be made for the subscript transformation functions for that array. Future research is needed to explore this conjecture.

For algorithms whose order of access does not depend on the content of the data, the order of access is almost trivial to model. The order of access of the remaining algorithms must he modelled stochastically. For example, the order of access of an edge following algorithm might be modelled by the statistics of a random walk. However, a complete model for the order of access to the memory is quite difficult is a multi-process environment.

Memory hierarchy might be adequately modelled by parameterizing each level of the hierarchy in terms of its granularity of storage, transfer latency time, and transfer bandwidth.

In the absence of specific models, a useful strategy for selecting the subscript transformation tables is to provide directionally isotropic time of access to regions of any size. This is important in order take advantage of characteristics the memory hierarchy, such as the ability to cluster many pages on the same track of the disk, and many tracks in the same cylinder to minimize head motion. The binary nested block scheme implements this strategy without needing a specific model for the memory hierarchy.

TRANSFORMATIONS VIA ACCESS TABLE MODIFICATION:

A useful number of geometric transformations on arrays can be accomplished by modifying the fi, fj access tables, without any modification of the array itself. These transformations include rotation by multiples of 90 degrees transposition, reflection, scaling, and translation. For instance, transposition is achieved by the exchange of tables between subscripts. reflection by reversing the content of the table, and rotation by a combination of transposition and reflection. Scale change and translation are accomplished by performing a linear mapping on the reference to the table entries. Is is improper to linearly map the actual values of the table entries.

IMPLEMENTATION:

Good programming practice requires that algorithms should not have imbedded in them unnecessary information about the objects which they manipulate. This notion, which is often called encapsulation, suggests that the details the storage of arrays should be of no concern to an algorithm which manipulates an array. Hence, the following functional forms of access is recommended:

```
value  = get-element(array, i, j, ...)

put-element(array, i, j, ... , value)

boolean := is-element(array, i, j, ...)
```

Is-element returns true is the subscripts are within the bounds of the array.

Such arrays are constructed by the function

```
array := new-array(type,size,ilb,iub,jlb,jub, ...)
```

This function allocates an array with subscript bounds determined by ilb, iub, ... using an implementation dependent default strategy to construct the tables. Type and size determine the data type and number of bits of the array elements.

An alternate form of array construction for user specified transformation tables is:

```
array := new-array(type,size,i-table,j-table,...)
```

The bounds of the array are determined by from the array bounds in the user specified tables.

The bounds of an array can be accessed by:

```
get-bounds(array, size,ilb,iub,jlb,jub, ...)
```

where ilb iub, ... are output parameters.

Input and output of such arrays is accomplished by:

```
array := get-array(file-name, mode)
```

Get-array constructs and returns an pointer to allow access to the array file on file-name. Mode specifies the permissible accessmode (read-only or read-write) for references to the array. It is assumed that no pages of array data are actually moved between virtual memory and the file until requested by element accesses to those pages. Therefore the size of the array does not significantly affect the time to access an element.

```
put-array(file-name, array, copy)
```

Put-array has the semantic effect of creating an array file named file-name which corresponds to the data in the array. The underlying implementation details may vary, but in the cases where the array is mapped to a temporary file. the parameter copy determines whether the data in the array is actually to the new file rather then renaming the temporary file.

DISCUSSION:

The proposed additive decomposition scheme using tables to evaluate the functions fi and fj make the implementation very efficient using conventional computers. When the array accesses are made in an order inappropriate to conventional array storage order, this scheme requires far less time than for conventional array accessesing schemes, otherwise the accessing times are comparable (see appendix III). This scheme also allows for the efficient access to images much larger than the virtual space of the machine.

The choices for these transformation functions which maximize the memory hit ratio for depend on the characteristics of the memory hierarchy of the computer system and the order of accesses to the elements of the array. It is conjectured that there are easily obtained models for system and algorithm access characteristics, from which a pragmatically optimum choice can be made for the subscript transformation functions.

A number of useful geometric transformations, such as rotation, transposition, translation, and scaling, can be performed by modifying these tables, and require no modifications to the storage of the array.

The semantics of a set procedures for array access, array creation, and the association of arrays with file names were defined. For computer systems with insufficient virtual memory, such as the PDP-10, a software virtual to phyaical mapping scheme is used. Implementations for the VAX and PDP-10 series computers to access pixels of large images stored as 2-dimensional arrays of n bits per element were presented in appendix III. Programs written in conformance with the proposed set of image primitives will be machine and operating system independent and will thereby greatly facilitate the interchange of image understanding programs.

REFERENCES

Reddy, Raj and Greg Gill (1978). "Representation Complexity of Image Data Structures", Proceedings: Image Understanding Workshop, May 1978, pp 28-30

APPENDIX I.

Implementation Semantica

This appendix contains a functional specification written in a mixture of ADA and engliah for a package of procedures for 2-dimensional array acceaa to arrays of a aingle, fixed type. Appendix II containa a more complete example of how a subroutine package might be

implemented. The intent of this specification is to define an implementation independent set of procedures and semantics for access to large arrays.

```
TYPE subscript IS INTEGER;

TYPE bigsubscript IS LONG INTEGER;

TYPE x_type IS  -- the type of the array elements

TYPE table_record IS
 RECORD
   lb,ub : subscript
        -- lower and upper bounds for table
  element  ARRAY (lb .. ub) OF bigsubscript,
 END RECORD

TYPE table IS ACCESS table_record;

TYPE x_type_array_record IS
 RECORD itab     : table;-- i transformation table
        jtab     : table;-- j transformation table
        elementtype  CONSTANT x_type_id;
        elementsize: CONSTANT x_type'SIZE;
                -- number of bits per element
--
The remaining fields are implementation dependent.
For a virtual memory implementation a pointer to an
array of x-type is defined.  For a software mapped
implementation a page table of pointers to arrays of
x-type are defined.
--
 END RECORD;

TYPE x_type_array IS ACCESS x_type_array_record;

FUNCTION getelement(    p    : IN x_type_array;
                        i,j  IN subscript)
        RETURN x;

PROCEDURE putelement(   p       : IN x_type_array;
                        i,j     : IN subscript;
                        val     : IN x)

FUNCTION iselement(     p  : IN x_type_array;
                        i,j: IN subscript)
       RETURN BOOLEAN IS
       RETURN
           i IN RANGE p.itab.lb .. p.itab.ub
       AND j IN RANGE p.jtab.lb .. p.jtab.ub


FUNCTION newxarray(
       si,sj   : IN subscript;
               -- number of columns and rows
       itab, jtab : IN table := NULL)
   RETURN x_type_array;
```

This function allocates an x_array with dimensions si, sj. The optional transform tables allow the user to specify his own coordinate transform strategy, otherwise an CREATETRANSFORM uses implementation dependent default strategy to construct the tables. NEWXARRAY calls CHECKTRANSFORM to check the range values of user defined itab and jtab transform tables to guarantee that all possible accesses using them will be within the bounds of the x_array.

```
type ACCESSMODE is (readonly, writeonly, readwrite);

FUNCTION xarrayinput(
        filename : STRING;
        mode: assessmode := readonly)
   RETURN x_type_array;
```

xarrayinput constructs and returns an xarraypointer to allow access to the array file on filename. Mode specifies the permissible accessmode for references to the array. It is assumed that no pages of array data are actually moved between virtual memory and the file until requested by element accesses to those pages. Therefore the size of the array does not significantly affect the time to access an element.

```
PROCEDURE xarrayoutput(p        : x_type_array;
                       filename : STRING;
                       copy     : BOOLEAN :=FALSE);
```

XARRAYOUTPUT has the semantic effect of creating an array file named filename which corresponds to the data in array p. The underlying implementation details may vary, but the following effects are intended:

if p resides in memory, then p is written into an ARRAY file

if p resides on a temporary file and copy=false then the temporary file is renamed to filename.

if p resides on a non-temporary file or copy=true then an exact copy of p is created on filename.

Xarray File Representation:

The precise file format is implementation dependent, but must contain sx, sy, xtypecode, xtab, ytab, and all of the pages of the array data. The following format is recommended:

A word is defined to 32 or 36 bits depending on the host computer. Pgsiz is the length of a virtual memory page on the host computer. A pointer within file is relative to the start of the file. A pointer within subfile is relative to word 0 of the subfile.

```
subfile
word
number
------
0: unique identifier for this subfile format
1: pointer within file to another subfile
2       sx number of columns
3: sy number of rows
4  xtypecode - a unique code identifying the
        data type x
5: xtypesize - the number of bits per element
        in data type x
6  pointer within subfile to the start of xtab
7  pointer within subfile to the start of ytab
8: pointer within subfile to the first page of
        array data
```

Appendix II

ADA DEFINITIONS FOR 2-D ACCESS TO LARGE ARRAYS

```
TYPE subscript IS INTEGER;

TYPE bigsubscript IS LONG INTEGER;

TYPE x_type IS  -- whatever type you want here for
                -- the array elements

TYPE table_record IS
 RECORD
   lb,ub : subscript;
        -- lower and upper bounds for table
  element: ARRAY (lb .. ub) OF bigsubscript;
 END RECORD

TYPE table IS ACCESS table_record;


TYPE x_type_array_body IS ARRAY (bigsubscript) of x

FOR x_type_array_body USE PACKING;
        -- to maximize storage efficiency

TYPE xarraybodyrecord IS
 RECORD size : bigsubscript;
        element : x_type_array_body(0 .. size);
 END RECORD;

TYPE xarraybodyrecordpointer IS
        ACCESS xarraybodyrecord;

xarraypagesize : CONSTANT 512; -- for example

TYPE xarraypage IS
 RECORD
  data : x_type_array_body (0 ... xarraypagesize-1);
 END RECORD;

TYPE xarraypagepointer IS ACCESS xarraypage;

TYPE x_type_array_record IS
 RECORD itab     : table:-- i transformation table
        jtab     : table:-- j transformation table
        elementtype. CONSTANT x_type_id;
        elementsize: CONSTANT x_type'SIZE;
                 -- number of bits per element
        virtual : BOOLEAN:
                -- TRUE=> virtual implementation
        CASE virtual OF
         WHEN TRUE =>
          data: xarraybodyrecordpointer;
                    -- this a pointer to the data
         WHEN FALSE=>
          numpages: INTEGER;
          pagesize: CONSTANT INTEGER xarraypagesize;
                -- for use by atorage manager
          extdata : filedescr;
                -- a record structure for the file
                -- holding the data
          pagemap : array (1 .. numpages) of
                        xarraypagepointer;
        END CASE;
 END RECORD;

TYPE x_type_array IS ACCESS xarrayrecord;
```

```
FUNCTION getelement(    p   : IN x_type_array;
                        i,j : IN subscript)
          RETURN x IS
BEGIN
  elenum: bigsubscript;
  pagnum : INTEGER:
  elenum :=  p.itab.element(i) + p.jtab.element(j);
  CASE p.virtual OF
    WHEN TRUE => RETURN(p.data.element(elenum));
    WHEN FALSE=>
        pagnum := 1 + elenum / pagesize;
        elenum := elenum MOD pagesize;
        IF p.pagemap(pagnum) = NULL THEN
              pagefault(p,pagnum);
        END IF;
        RETURN(p.pagemap(pagnum).data(elenum));
  END CASE:
END;

PROCEDURE putelement(   p       : IN x_type_array;
                        i,j     : IN subscript;
                        val     : IN x) IS
BEGIN
  elenum: bigsubscript;
  pagnum : INTEGER:
  elenum :=  p.itab.element(i) + p.jtab.element(j);
  CASE p.virtual OF
    WHEN TRUE => RETURN(p.data.element(elenum));
    WHEN FALSE=>
        pagnum := 1 + elenum / pagesize;
        elenum := elenum MOD pagesize;
        IF p.pagemap(pagnum) = NULL THEN
              pagefault(p,pagnum);
        END IF;
        p.pagemap(pagnum).data(elenum) := val;
  END CASE:
END;
```

The procedure PAGEFAULT is not defined here, since it will normally be implementation dependent. The intent is that space will be found large enough to store one xarraypage, and a corresponding xarraypagepointer will be stored into p.pagemap(pagenumber). Usually, this will require searching a global table of pages to find either a free page, or the least-recently-used page which will be "kicked-out".

APPENDIX III

Machine Specific Implementations of Image Access

We now consider implementations of this scheme for both the VAX and the KL-10, with two different assumptions: whether the entire image will or will not fit into an acceptable fraction of the computer's virtual memory apace.

The implementer is encouraged to expend maximum effort to make the two element acceas functiona GETELEMENT and PUTELEMENT as efficient aa possible, either through machine coded subroutines, in-line code generation from within the host programming language, or code gsneration by the NEWXARRAY generation procedure. If they are efficiently implemented, then researchers developing new algorithms will not be tempted to

resort to special representations or coding tricks just to get more efficiency, and the resulting programs will be clearer and more transportable to any sites which implement these semantics. A proper implementation of these access primitives should be as efficient as most host languages will produce for standard 2-d array access.

We will first consider a VAX virtual memory implementation for variable pixel sizes. A MAINSAIL language definition of pixel access using the VAX variable bit field manipulation primitives could be written as:

```
INTEGER PROCEDURE getpix(
        POINTER(image) p:
         integer x,y):
        RETURN(extract-field(
                p.bpp*(p.xtab[x]+p.ytab[y]),
                p.bpp, p.picbase)):

PROCEDURE putpix(
        POINTER(pix) p;
        INTEGER x,y,v):
        insert-field(v,
                p bpp*(p.xtab[x]+p.ytab[y]),
                p.bpp, p.picbase):
```

where extract-field and insert-field correspond to the VAX variable bit field instructions.

A pair of very efficient VAX assembly language code sequences to access pixels using the MAINSAIL record storage conventions can be written as follows. These sequences assume that the pointer to the picture record is in register RP, and the image coordinates are in registers RX and RY respectively.

```
GETPIX: ADDL3 @XTABO(RP)[RX],@YTABO(RP)[RY],RTMP
         ; computes pixel number in image
        MULL2 BPP(RP),RTMP
         : computes bit offset
        EXTZV RTMP,BPP(RP),PICBASE(RP),R1
         : extract the pixel into R1
```

Note that XTABO and YTABO are offsets into the IMAGE record which contain the virtual origins of the XTAB and YTAB tables. That is, the longword at XTABO+X corresponds to XTAB[X]. Also note that only one ADDL3 instruction is needed to compute the pixel offset.

```
putpix: ADDL3 @XTABO(RP)[RX],@YTABO(RP)[RY],RTMP
         ; pixel number in image
        MULL2 BPP(RP),RTMP
         ; compute bit offset
        INSV RVAL,RTMP,BPP(RP),PICBASE(RP)
         : insert the pixel value in RVAL
```

The instructions shown in these procedures require only 8 data references plus 18 bytes of program reference. The overhead of a CALLS type procedure call is certain to be more costly than the pixel access itself. Consequently, it is advised that either in-line code be generated for pixel access, or that a JSB type procedure call passing arguments in registers be used.

The MULL2 multiply instruction can be eliminated by multiplying the values stored in the xtab and ytab tables by bpp as follows:

```
xtab[x] _ xtab[x] * bpp
ytab[y] _ ytab[y] * bpp
```

This scheme has been tested and requires about 9 microseconds per access with the MULL3 instruction and about 7 microseconds per access otherwise For accesses to byte, word or longword pixels the EXTZV instruction can be replaced with the appropriate MOV instruction. This reduces the pixel access time to about 4 microseconds.

DEC PDP-10 series software paged implementation:

The PDP-10 implementation is defined in SAIL. An image record class is declared as follows:

```
class pix(
        INTEGER sx;           # image width
        INTEGER sy;           # image height
        INTEGER bpp;          # bits per pixel
        INTEGER ARRAY xtab;   # x mapping table
        INTEGER ARRAY ytab;   # y mapping table
        INTEGER ARRAY ptrtab;# byte pointers TABLE
        INTEGER ARRAY pagtab;# the mapping table
        INTEGER pagsiz;       # words per page
        INTEGER fileid;       # disk file id
        INTEGER gtpix;        # entry to code
        INTEGER ptpix;        # entry to code
```

Each image has a page table in its image descriptor record which encodes for each page the following information:

```
pagtab[page] = 0      : page is not in memory
             = vpage : virtual address of page
```

Each image also has a table of bytepointers, one for each pixel in an image page. This table can be shared between all images with the same pixel size. This table is constructed as follows:

```
ptr_point(bpp,0,bpp-1) + 4 lsh 18:
```

This constructs a bytepointer to the first byte in the word indexed by register 4 with byte size bpp.

```
for i_1 step 1 until pagsiz do
 begin ptrtab[i] _ ptr: ibp(ptr);end:
```

where IBP increments the bytepointer to the next byte.

A procedure PAGEFAULT is used to manage the paging tables and perform the transfer of data between the virtual memory and the file system. It is assumed that the read and write access modes are enforced by the pagefault procedure. For brevity, PAGEFAULT is not defined in this document.

The optimal PDP-10 code sequence is obtained by generating a version of the getpix and

putpix procedures for each image using the following PDP-10 assembly language sequences:

```
; gtpix assumes the registers are setup as follows:
; ac1: x
; ac2: y
; ac5  pointer to pic record
; ac6: return address
; result is returned in ac1

GTPIX:  MOVE    4,XTABBASE(1)
        ADD     4,YTABBASE(2)    ; element number
        HLRZ    2,4              ; offset(see note 1)
        SKIPN   4,MAPTABBASE(4) ; get virtual addr
        PUSHJ   P,PAGEFAULT      ; page not there
        LDB     1,PTRTABBASE(2) ; get the pixel
                ; byte pointers index by AC4
        JRST    @6

; ptpix assumes the registers are setup as follows:
; ac1: x
; ac2: y
; ac3: value to store
; ac5  pointer to pic record
; ac6  return address
; result is returned in ac1

PTPIX   MOVE    4.XTABBASE(1)
        ADD     4,YTABBASE(2)    ; element number
        HLRZ    2,4              ; offset (note 1)
        SKIPN   4,MAPTABBASE(4) ; get virtual addr
        PUSHJ   P,PAGEFAULT      ; page not there
        DPB     3,PTRTABBASE(2) ; get the pixel
                ; byte pointers index by AC4
        JRST    @6               ; return
```

Note 1

For images which have a power of 2 pixels per page, the xtab, ytab tables can be constructed so that no divide is required to compute the page number and offset within page. This is accomplished by modifying the access tables as follows:

```
modified-tab[i] _ tab[i] DIV pagsiz     # page
                 +tab[i] MOD pagsiz LSH 18:# offset
```

These optimized procedures are attached to the image record fields GTPIX and PTPIX respectively, and are not to be called directly from SAIL, but instead from these procedures.

```
INTEGER PROCEDURE getpix(POINTER(pix) pic:
                         INTEGER X,Y):
START!CODE
        POP P,6: return address
        POP P,2: y
        POP P,1: x
        POP P,5  pic
        JRST @gtpixoffset(4)
END;

INTEGER PROCEDURE getpix(POINTER(pix) pic:
                         INTEGER x,y,val):
START!CODE
        POP P,6; return address
        POP P,3; val
        POP P,2; y
        POP P,1; x
        POP P,5; pic
        JRST @ptpixoffset(4)
END;
```

Both SAIL and MAINSAIL packages have been implemented for access to variable pixel size, software mapped image files. The following loop was used to determine the pixel access times:

```
FOR Y:=0 STEP 1 UNTIL 255 DO
 FOR X:=Y STEP 1 UNTIL 255 DO
        BEGIN   INTEGER V1,V2;
                GETPIX(PIC,X,Y,V1);
                GETPIX(PIC,Y,X,V2);
                PUTPIX(PIC,X,Y,V2)
                PUTPIX(PIC,Y,X,V1);
        END
```

The total CPU time on a KL-10 for a 256x256 image of 8 bits per pixel was 4 seconds, or about 8 microseconds per access including the loop overhead. Note that the GETPIX and PUTPIX functions used here were SAIL in-line macros rather than procedure calls.

The following loop which accesses the elements of a normal 2-d SAIL array in strictly storage order also required about 1 second to execute, or about 8 microseconds per access:

```
FOR K_1 STEP 1 UNTIL 4 DO
FOR I_0 STEP 1 UNTIL 256 DO
 FOR J_0 STEP 1 UNTIL 256 DO
        RESULT_A[I,J];
```

One concludes from these examples that if the pixel access functions are carefully implemented, then there is little or no cost in time over conventional array accesses.

/ 4W

SOME USES OF PYRAMIDS IN IMAGE PROCESSING AND SEGMENTATION

Azriel Rosenfeld

Computer Vision Laboratory, Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

This paper summarizes recent applications of multi-resolution ("pyramid") image representations in image analysis and processing, including

a) Approximation of Gaussian-kernel convolution operators by iterated local convolutions

b) Construction of overlapped pyramids to reduce shift sensitivity

c) Clustering of pyramid nodes into trees representing image regions

d) Use of local operations in a pyramid to detect blob-like and streak-like objects in the image

e) Use of pyramids to define sets of quadtree approximations to the image ("Q-images")

f) Image segmentation by analysis of the histograms of Q-images

g) Improved edge detection based on associations between edges in the image and a corresponding Q-image.

INTRODUCTION

"Pyramid" structures, that is, sequences of arrays of exponentially reduced resolution derived from a given image, have been used by a number of investigators [e.g., 1-6] for image processing and analysis. The simplest pyramid construction scheme assumes that the input image is 2^n by 2^n, and reduces resolution by a factor of 2 at each step using 2-by-2 block averaging, so that the pyramid has n+1 levels of sizes $2^n x 2^n$, $2^{n-1} x 2^{n-1}$, ..., 2x2, 1x1. This paper describes some generalizations of the pyramid construction process, and briefly presents some applications of pyramid structures to image processing and segmentation. Further details can be found in individual technical reports.

GAUSSIAN CONVOLUTIONS

In general, pyramids can be constructed using weighted averages over neighborhoods of any desired size, which can be allowed to overlap. If we require the weighting function to satisfy certain very natural constraints, it turns out that the weight patterns after a few iterations become almost exactly Gaussian. For simplicity, we first illustrate this for the case of a one-dimensional array f(x) and a weighting function of odd width (w(i) for $-m \le i \le m$).

Let $g_0(n) = f(n)$ and $g_\ell(nr^\ell) = \sum_{i=-m}^{m} w(i) g_{\ell-1}(nr^\ell + ir^{\ell-1})$ for $\ell \ge 1$; thus each g_ℓ is a weighted average of 2m+1 $g_{\ell-1}$'s spaced $r^{\ell-1}$ apart. We adopt the following simple constraints on the w's:

(1) $\sum_{i=-m}^{m} w(i) = 1$

(2) $w(-i) = w(i)$, $1 \le i \le m$

(3) $0 \le i \le j$ implies $0 \le w(j) \le w(i)$

(4) $\sum_{i=-m}^{m} w(k+ir) = \frac{1}{r}$, $0 \le k < r$

Constraints (2-3) require the w's to have a symmetric central peak, and constraint (4) insures that each f(n) (except near the array border) contributes with equal weight at every level ℓ. Evidently, each g_ℓ is the convolution of f with an "equivalent kernel" h_ℓ, obtained by expanding the definition of g_ℓ until it is expressed in terms of w's and f's.

For example, let m = 2, r = 2, and w(0) = a, w(1) = b, w(2) = c (see Figure 1a). Then constraints (1-4) yield $1/4 \le a \le 1/2$, $b = 1/4$, and $c = 1/4 - a/2$. Figure 1b shows the equivalent convolution kernels h_ℓ using a = 0.4 for ℓ = 1,2,3, and for large ℓ; note that the shape of h_ℓ rapidly approaches a "Gaussian" (the discrepancy from a Gaussian is in fact very small). The approximation to a Gaussian remains quite good for a in the range 0.3 to 0.45, with the Gaussian becoming more sharply peaked as a increases(Fig.1c).

Similar results can be obtained for even w and for non-integer r. The generalization to two dimensions is also straightforward, especially if we take the kernel w to be separable. By taking linear combinations of Gaussian convolutions, one can construct operators of many sizes that respond

to spots, edges, or bars. Gaussian convolutions can be computed in the pyramid more efficiently than with the FFT, typically by two or three orders of magnitude. For the details on all of these matters see [7].

OVERLAPPED PYRAMIDS

The general pyramid construction process described above allows the averaging neighborhoods to overlap--e.g., when m = r = 2, level 1 is constructed from level 0 using neighborhoods of width 2m + 1 = 5 spaced 2 apart, and similarly at higher levels. Note that in spite of the overlap, the pyramid still tapers exponentially.

An advantage of overlapped pyramids, as compared to the standard, unoverlapped block averaging scheme, is that when local feature detectors are applied at various levels of a nonoverlapped pyramid, their ability to detect large features in the image depends strongly on the positions of these features. For example, suppose we want to detect bloblike objects of diameter ≈4. If such an object is located in a position whose coordinates are multiples of 4, it will (approximately) be contained in one of the 4x4 blocks whose averages are at level 2 of the pyramid, and the neighboring 4x4 blocks will be (approximately) disjoint from it, so that a local spot detector applied at level 2 will respond to it. On the other hand, if the coordinates are odd multiples of 2, the object overlaps four adjacent 4x4 blocks, whose averages will respond only weakly to its presence. This position dependence is reduced if we use an overlapped pyramid--e.g., in one dimension, if the values at successive layers represent averages of 50% overlapped intervals:

Level	Intervals
1	[1,2]; [2,3]; [3,4]; ....
2	[1,4]; [3,6]; [5,8]; ....
3	[1,8]; [5,12]; [9,16]; ....

Here a spot detector would be based on adjacent, nonoverlapping blocks, e.g. [5,8] vs [1,4] and [9,12]; and the center block has stronger response than both adjacent blocks even if the object is not in the optimal position. Note that in this simple example of an overlapped pyramid, level 1 is nearly the same size as level 0, but the exponential shrinking begins at level 2.

REGION EXTRACTION BY PYRAMID NODE CLUSTERING

A number of image smoothing algorithms have been developed in which each pixel is averaged with a subset of its neighbors chosen so as to make it likely that the neighbors in the subset all belong to the same image region as the given pixel; e.g., we can use the neighbors whose gray levels most resemble that of the pixel, or we can examine a set of asymmetric neighborhoods of the pixel and choose one whose average gray level most resembles that of the pixel. It would be of interest to generalize this concept to a pyramid environment; this might involve associating nodes at a given level with "neighboring" nodes at the next higher level that appear to belong to the same region, and so on. If this could be done, the averaging process could propagate across a region much more quickly (in a number of iterations proportional to the log of the region diameter), as the largest blocks contained in the region become associated with the smaller blocks near the region's borders.

The following is a simple scheme for linking blocks in an overlapped pyramid (based on 4x4 blocks with 50% overlap) so as to obtain linked clusters of blocks representing two or three types of regions. The pyramid is initialized by performing the successive overlapped 4x4 averages. At the top level, which we take to be 2x2, the resulting four averages are grouped into two or three classes, depending on the desired number of regions and on their expected relative sizes, and each group is given a single average value. Each block B, say at level ℓ, contributes to the averages of several (overlapped) "father blocks" at level ℓ+1--four, in our experiments; we now link block B to that father block (or group, if B is just below the top level) whose average is closest to B's. Next, we recompute the block averages, with each block at level ℓ+1 using only the averages of the blocks at level ℓ linked to it to compute its new average. Since this changes the block averages, the links may now change, so we can repeat the entire process. Typically, the process stabilizes after 10 to 20 iterations; at that stage, the linked sets of blocks define a good decomposition of the image into regions.

Figure 2a shows an example (a 64x64 FLIR image of a tank) in which, at the 2x2 level, three of the averages were grouped together, while the fourth one defined its own class. The original pyramid is shown at the lower left, while the upper nine images show the average gray level of the group to which each pixel is linked at iterations 1, 2, 3, 4, 5, 6, 7, 14, and 26. Note that at iteration 1 the tank is nearly invisible, since the averages at the 2x2 level are initially nearly the same. The results are sensitive to the number of classes used at the 2x2 level, and to the sizes of these classes; Figure 2b shows corresponding results when the four averages at the 2x2 were grouped into two classes of two each (the "tank" class is too large), and Figure 2c shows results when three classes were used at the 2x2 level. The results are also sensitive to noisiness in the original image; in Figures 2a-c, the input image was blurred before building the pyramid, and Figure 2d shows what happens when this is not done (in the case where two groups of two each are used at the 2x2 level). Finally, Figure 2e shows a 3-class result for a picture of a blood cell; it yields a good segmentation into nucleus, cell body, and background. Further experiments with this approach are in progress, and will be documented in a forthcoming technical report.

BLOB AND STREAK DETECTION

Pyramid representations provide a method of using simple types of shape information in the segmentation process, rather than first segmenting and then editing the segmentation to yield regions of the desired shapes. For example, if we want to extract blob-like (compact) objects, we can apply local spot detection operators at each level of the pyramid (if the size of the desired objects is known, we can restrict this to a few levels), and bias the segmentation process to favor object extraction in those parts of the image corresponding to the positions of detected spots. A simple way of doing this [8] is to define a local threshold at the position of each detected spot, e.g. midway between the average gray levels of the spot's center and surround; this threshold should be effective for extracting the object at that position.

An analogous approach can be used to favor the extraction of streak-like objects, i.e., linear features of arbitrary thickness. Such objects give rise to thin lines at appropriate levels of the pyramid. Results are improved if a local line enhancement process is applied to the line detection outputs at each pyramid level. Figure 3 shows the results of applying this process to portions of aerial photographs showing several Maryland airports. These results were obtained with a non-overlapped pyramid; they improve when an overlapped pyramid is used. A further possibility for improvement is to use the pyramid line detections to initialize linear feature tracking processes in the image, rather than simply thresholding.

QUADTREE APPROXIMATIONS TO AN IMAGE

The quadtree representation of a binary image (of size $2^n \times 2^n$) is constructed by repeated subdivision into quadrants; a block is subdivided unless it consists entirely of 1's or of 0's. More generally, given a criterion for the homogeneity of a block, we can use a similar process to construct quadtree approximations to an image, by subdividing blocks into quadrants unless they are homogeneous [4]. For example, suppose that we call a block homogeneous if its standard deviation is less than some threshold, and if so, we represent the block by its mean. When we construct a pyramid representation of an image by the standard 2x2 averaging process, we can compute the standard deviation as well as the mean of each block; for any threshold t, we thus have implicitly defined a quadtree approximation to the image, where a block is regarded as a leaf node of the tree if its standard deviation does not exceed t.

Defining homogeneity in terms of the standard deviation gives us a zero-order piecewise least-squares approximation to the image, since the constant which best approximates an image block in the least squares sense is the block's mean, and the corresponding least-squares error is the variance. More generally, we can define piecewise least-squares approximations by polynomials of degree k, for k = 1, 2, ..., and call a block homogenous if the error in its degree-k least-squares approximation does not exceed a given threshold; this allows us to define higher-order quadtree approximations. (Haralick [10] has shown that first-order approximations have significant advantages over zero-order approximations for purposes such as image smoothing and edge detection.)

The least-squares polynomial approximations (of any given degree) to an image can be computed hierarchically. If we know these approximations, and their associated errors, on the subblocks of a block, we can compute the approximation and its error for the entire block directly from this information, without examining the original image gray levels. This allows us to construct higher-order quadtree approximations "bottom-up", starting from small blocks (e.g., for which the approximation of the given degree is exact), and computing the approximations for larger blocks until a given error threshold is exceeded. The details of this procedure can be found in [11].

QUADTREE APPROXIMATIONS AS AIDS TO THRESHOLDING

Suppose we have constructed a zero-order quadtree approximation to a given image; this approximation, which we call a "Q-image", consists of a set of blocks (corresponding to the leaves of the quadtree) each having constant gray level (the mean gray level of the corresponding image block, which had a below-threshold standard deviation, and so can be approximated by its mean).

If the image consists of homogeneous objects on a homogeneous background, the Q-image should contain large blocks in the interiors of the objects and background, as well as smaller blocks near their borders. The image histogram should consist of two peaks representing the object and background gray level populations, but these peaks may overlap due to the existence of intermediate gray levels in the border zones. In the histogram of the Q-image (for brevity: the Q-histogram), the peaks should be sharper, since averaging reduces gray level variability; this expectation is confirmed by the example in Figure 4. Note that the Q-histogram can be computed rapidly from the quadtree approximation, by counting each leaf node a number of times equal to its area.

Further improvement in the Q-histogram can be obtained by suppressing the contributions of small blocks; this tends to eliminate border gray levels while retaining levels that lie interior to the objects or background. Conversely, if we consider the small blocks only, we should obtain a unimodal histogram representing border gray levels, and the mean of this histogram should be a good threshold for separating the objects from their background. Examples of the results of suppressing small blocks or using only small blocks are shown in Figure 4 [12].

More generally, if there are several peaks in the Q-histogram, they should correspond to different types of homogeneous regions in the image; note that because of the peak sharpening effect, such peaks will be easier to detect in the Q-histogram than in the original histogram. Whenever we detect such a peak, we can find the blocks in the Q-image that give rise to it, and histogram the image gray levels in the vicinity of these blocks; this allows us to define a local threshold suitable for extracting the regions represented by the peak. If desired, the threshold can be adjusted to yield a good match between the borders of the extracted regions and the edges in the image, as in the SUPERSLICE algorithm (but without the need for connected component analysis). Some examples of images segmented in this way are shown in Figure 5 [13].

QUADTREE APPROXIMATIONS AS AIDS TO EDGE DETECTION

If we apply an edge operator to a Q-image, we obtain edges that are generally stronger than those in the original image (since the regions on the two sides of the edge have reduced gray level variability because of the averaging), but that are displaced from their proper positions, since they are constrained to lie on the cracks between blocks. (It should be pointed out that edge operators can be applied directly to the quadtree, using neighbor-finding techniques to determine the gray level differences between adjacent pairs of blocks; this should be much less costly than constructing a full-resolution Q-image and applying edge operators at every pixel.) If we can associate these Q-edges with edges in the original image, it should be possible to obtain edge maps representing significant edges (as obtained from the Q-image) in accurate positions (as obtained from the original image).

Several methods of associating edges with Q-edges are investigated in [14]; the details will not be given here. Figure 6 shows an example using an iterative enhancement process, based on the Q-edges, to selectively enhance edges on the original image, with very good results.

CONCLUDING REMARKS

The results in this paper illustrate a few of the ways in which pyramid representations of images can be useful in image processing and analysis. Further work in this area is in active progress.

REFERENCES

1. L. Uhr, Layered "recognition cone" networks that preprocess, classify, and describe, IEEETC-21, 1972, 758-768.
2. S. L. Tanimoto and T. Pavlidis, A hierarchical data structure for picture processing, CGIP 4, 1975, 104-119.
3. S. L. Tanimoto, Pictoral feature distortion in a pyramid, ibid. 5, 1976, 333-352.
4. T. Pavlidis, Structural Pattern Recognition, Springer, New York, 1977.
5. E. M. Riseman and M. A. Arbib, Computational techniques in the visual segmentation of static scenes, CGIP 6, 1977, 221-276.
6. A. R. Hanson and E. M. Riseman, Segmentation of natural scenes, in A. R. Hanson and E. M. Riseman, eds., Computer Vision Systems, Academic Press, New York, 1978.
7. P. J. Burt, Fast, hierarchical correlations with Gaussian-like kernels, TR-860, Computer Vision Laboratory, University of Maryland, College Park, MD, January 1980.
8. M. Shneier, Using pyramids to define local thresholds for blob detection, TR-808, Computer Vision Laboratory, University of Maryland, College Park, MD, September 1979; also in Proc. Image Understanding Workshop, November 1979, 31-35.
9. M. Shneier, Extracting linear features from images using pyramids, TR-855, Computer Vision Laboratory, University of Maryland, College Park, MD, January 1980.
10. R. M. Haralick, Edge and region analysis for digital image data, CGIP 12, 1980, 60-73.
11. P. J. Burt, Hierarchically derived piecewise polynomial approximations to waveforms and images, TR-838, Computer Vision Laboratory, University of Maryland, College Park, MD, November 1979.
12. A. Y. Wu, T. H. Hong, and A. Rosenfeld, Threshold selection using quadtrees, TR-886, Computer Vision Laboratory, University of Maryland, College Park, MD, March 1980.
13. S. Ranade, A. Rosenfeld, and J. M. S. Prewitt, Use of quadtrees for image segmentation, TR-878, Computer Vision Laboratory, University of Maryland, College Park, MD, February 1980.
14. S. Ranade, Use of quadtrees for edge enhancement, TR-862, Computer Vision Laboratory, University of Maryland, College Park, MD, February 1980.

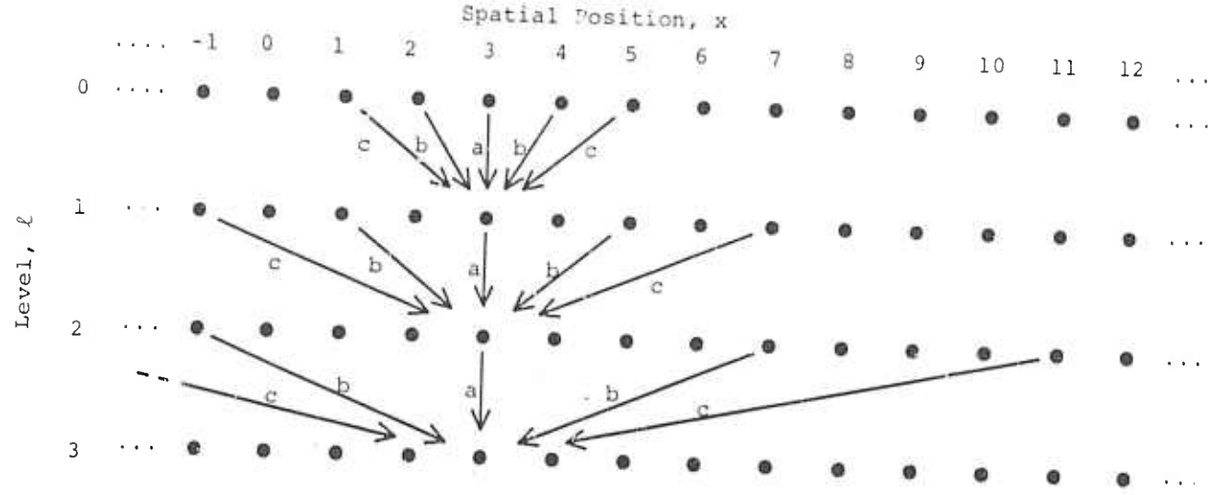


Figure 1a. Weighting scheme for width-5 neighborhood and scale factor of 2.

(Figure 1b. on next page)

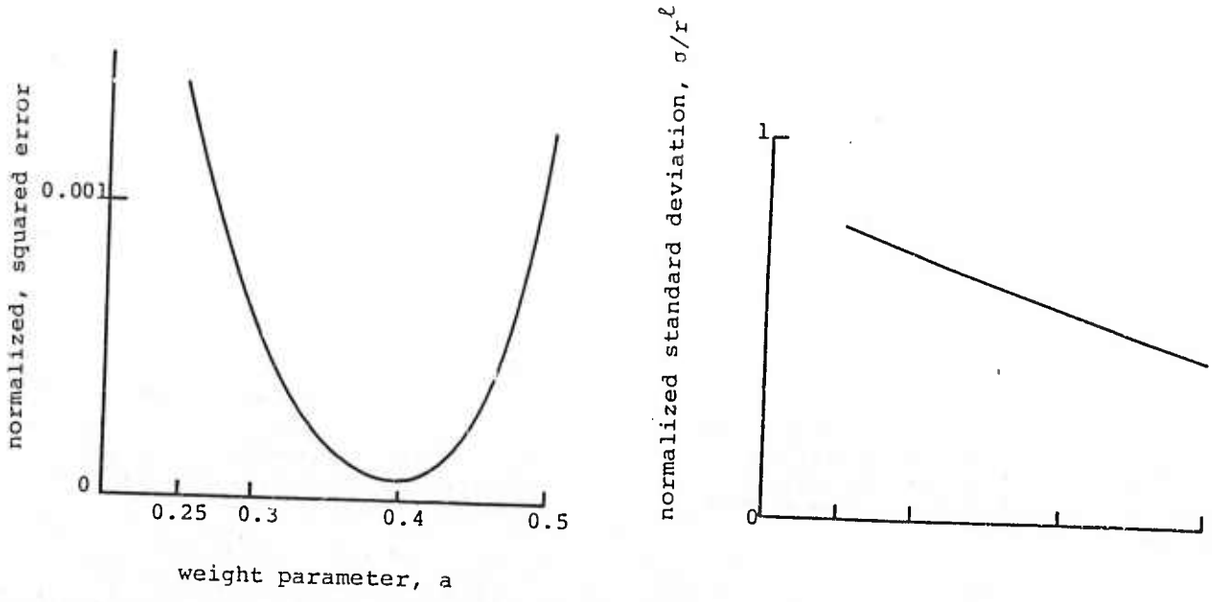


Figure 1c. Discrepancy from Gaussian, and standard deviation of Gaussian, as functions of parameter a.

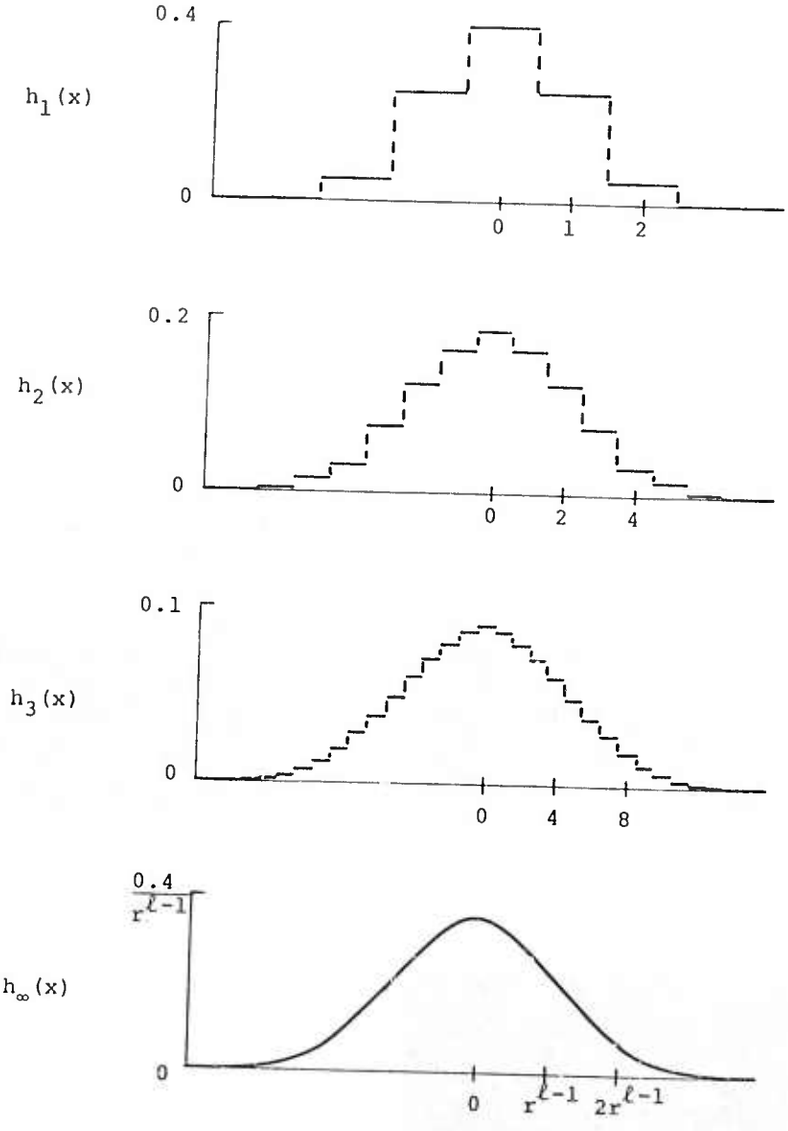
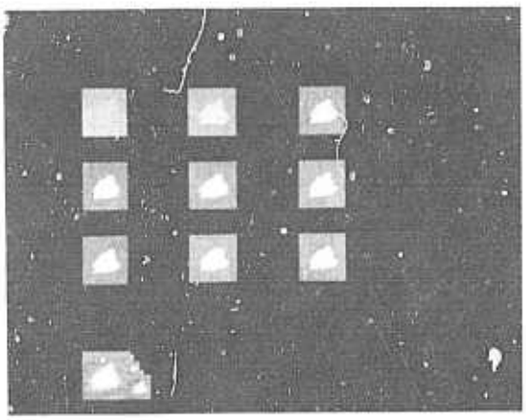
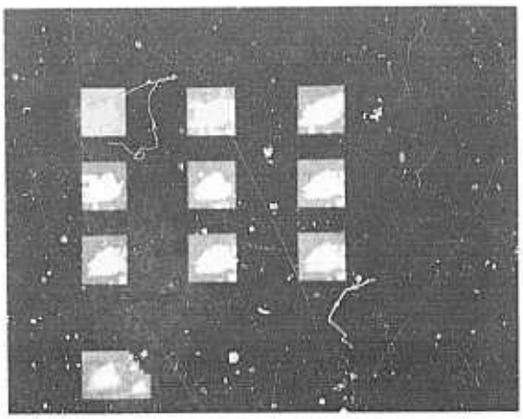


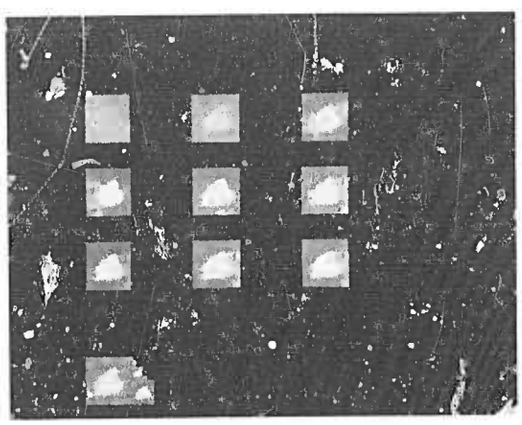
Figure 1b. Resulting equivalent convolution kernels at successive levels.



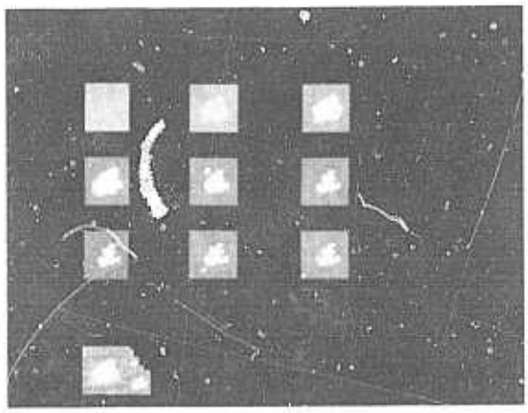
a.



b.



c.



d.

e.

Figure 2. Region extraction by pyramid node clustering.
a) Results for a FLIR image of a tank (see text) when the averages are grouped (3,1) at the 2x2 level.
b) Results for (2,2) grouping
c) Results for (2,1,1) grouping
d) Results for (2,2) grouping when the input image is not blurred
e) Results for a blood cell image using (2,1,1) grouping

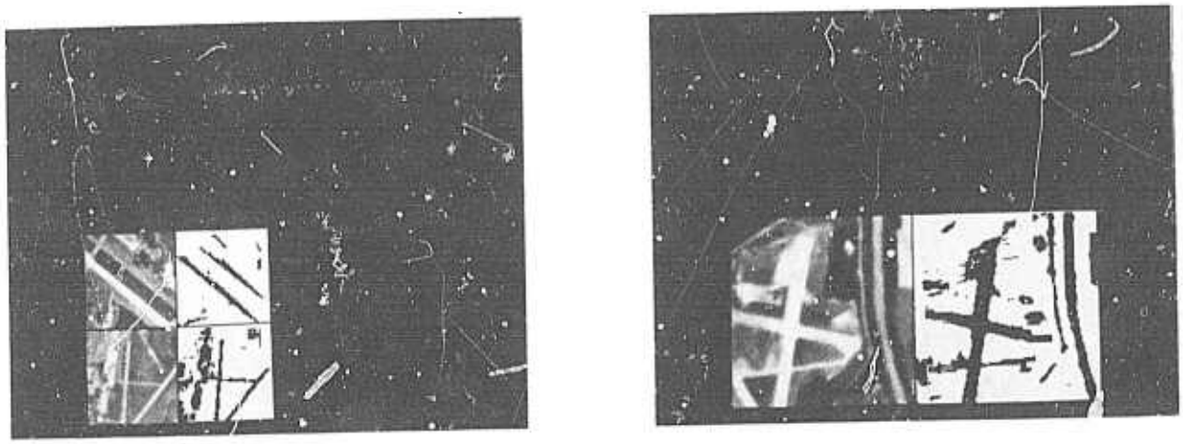


Figure 3. Pyramid streak extraction: Results for three airfield images.

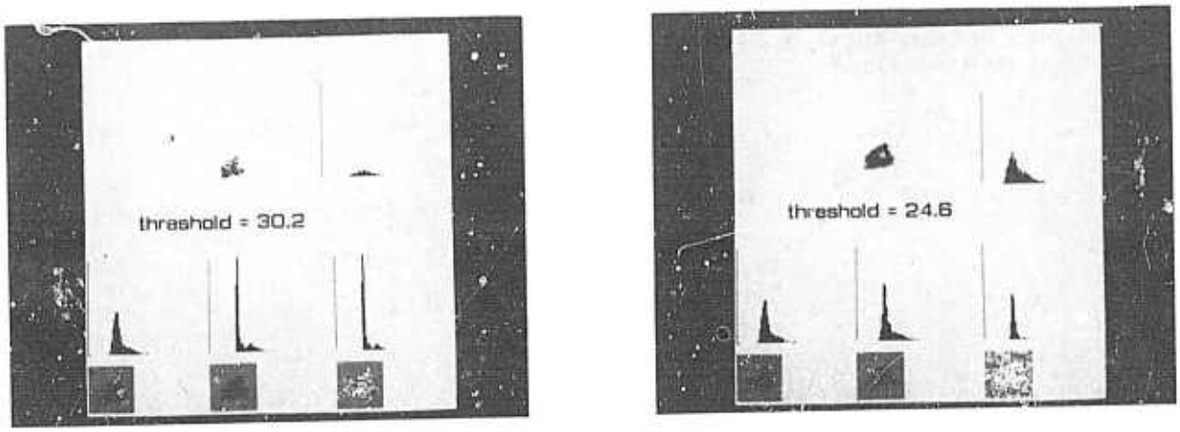


Figure 4. Q-images as aids in thresholding. The standard deviation tolerances in the two examples are 2.6 and 1.6, respectively. Counterclockwise from left:
Original image and histogram
Q-image and histogram
Q-image with small leaves deleted, and histogram
Histogram of small leaves only
Result of thresholding at mean of small-leaf histogram

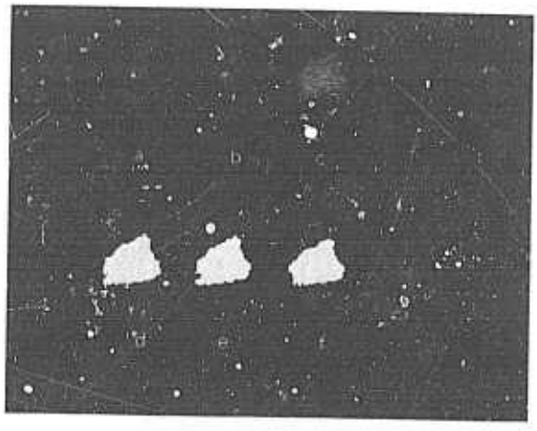


Figure 5. Segmentation with the aid of a Q-image.
a) Result of applying Prewitt edge operator to a tank image
b) Cleaned edge map (see Figure 6)
c) Blocks belonging to the tank peak on the Q-image
d-f) Regions extracted by local thresholding in the vicinity of these blocks, for three choices of the threshold.

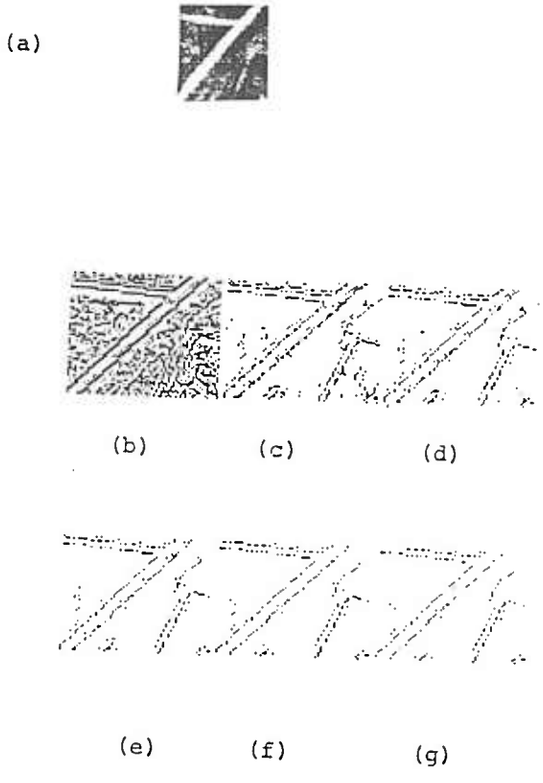


Figure 6. (a) Image of a part of Frederick Airport.
(b) Result of non-maximum suppression applied to the Prewitt edge detector for 6(a) thresholded at 0.
(c) to (g) Results of successive iterations of the enhancement process thresholded at 0.

SOLVING FOR THE PARAMETERS OF OBJECT MODELS FROM IMAGE DESCRIPTIONS

David G. Lowe

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

A solution is given to the problem of calculating the parameter values needed to bring the projection of a three-dimensional object model into correspondence with an image. The predictive components of the ACRONYM system form tentative matches between elements of the model and elements of the image description. These initial matches are used to determine values for projection and model parameters as a step towards making more detailed predictions. The solution uses Newton-Raphson convergence in conjunction with a modified camera transform which is expressed in terms of image-centered parameters. The solution is able to use lines in an image even when we lack knowledge of their terminations. The method can also be used to determine the positions of articulated parts of models and can handle constraints on the object position.

1. Introduction

Any recognition task requires us to first have a representation for the objects which are to be recognized. For the visual recognition of three-dimensional objects, the most obvious and natural representation is probably the use of three-dimensional models which give a full specification of an object's shape and appearance in a viewpoint independent manner. The field of computer graphics has already had a considerable amount of experience in working with such models, but within the field of computer vision models of this sort have rarely been used. One difficulty encountered in using these models for computer vision has been the problem of calculating the position and orientation of the model which will bring its projection into correspondence with the image. This paper describes a solution to this problem which makes use of only a few tentative initial matches between the model and image. In addition, the three-dimensional model need not be fully specified and may contain articulated parts. We show how the values of these model parameters can be derived from the image at the same time that we calculate the object's projection parameters. We refer to these two issues as the problem of back-projection, since we must infer the projection and model parameters from the image, rather than the other way around. These new techniques should make it much easier to use three-dimensional models in computer vision applications and derive the many other benefits of their use.

Obviously, the techniques of back-projection are not sufficient in themselves to implement a complete vision system. The research described here is just one part of the ACRONYM research project at Stanford [2, 3, 4], which brings together the many other components required in a functioning visual recognition system. ACRONYM includes both interpretive (bottom-up) and predictive (top-down) components. The following discussion outlines three major problems which have been inhibiting the use of three-dimensional models in computer vision, and describes the techniques used within ACRONYM to overcome them:

(1) One of the most difficult problems in the recognition process is in getting started. It is hard to see how to make any initial use of a viewpoint independent model if we have no prior information about its orientation or location in the image. This problem is handled within ACRONYM by a general geometric reasoning system for deriving sets of quasi-invariant features from the model. These quasi-invariants are features which are observable over a wide range of orientations and viewing conditions (for example, parallel lines in the model will appear as parallel lines in the image under most viewing conditions, and there will be restrictions on their relative positions). By creating several sets of quasi-invariants for an object and searching the image for instances of them, it is possible to set up hypothesized matches between a few image features and parts of the model to provide a starting point for the techniques described in this paper. The quasi-invariants are used only to generate hypothesized matches, and not to evaluate the correctness of a match, so it is quite reasonable for this first stage to return many

incorrect matches for each correct one.

(2) A second problem with using three-dimensional models has been the difficulty of mathematically manipulating them in order to compare them to features in the image. It is largely with this problem that this paper is concerned. The usual techniques of computer graphics are unintuitive and inappropriate when applied to the back-projection problem, and here we develop new mathematical tools for deriving the projection parameters of the model present in a particular image.

(3) Another important objection to most computer graphics modelling systems is that the models seem to be too tightly specified. We often wish to represent looser constraints than giving the exact three-dimensional coordinates of all parts of the model. This problem is partially solved in ACRONYM by allowing almost all aspects of the model to be parameterized in terms of variables, and an important part of the back-projection solution given here is the ability to solve for the values of these variables in a specific image. ACRONYM also uses a system of tree-structured constraints which allows the specification of generic classes of objects, with the further specification of specific instances in terms of the original generic model [2].

Computer vision researchers have rightly been wary of investing the large amount of time and effort necessary to create a full geometric modelling system when it has not been easily apparent how to use these models. We hope that the techniques given in this paper will show that such models can be used in useful ways. For many recognition tasks, there is no clear alternative to the use of viewpoint-independent models. Even when an alternative does exist, an important advantage in the use of three-dimensional models is that they allow us to do the matching in the domain of volumes rather than the domain of images, which gives the matching process a much more appropriate basis on which to make comparisons and measure error. In particular, the ACRONYM system is able to work effectively within this domain because all our models are themselves built up from primitive volume elements in the form of generalized cones [1], rather than using the more common surface descriptions of objects. Generalized cones also provide a particularly appropriate parameterization for many objects. However, the techniques outlined in this paper are applicable to almost any three-dimensional representation.

2. Forward Projection

Before the back-projection techniques for deducing the projection parameters can be presented, it is first necessary to understand the methods used for projection in the forward direction. The projection method presented here is similar to those which are commonly used for computer graphics [6]. In essence, the technique is to specify the type of camera being used and its location and orientation with respect to the three-dimensional model. These parameters are used in a coordinate transform to compute two-dimensional coordinates for points in the image from three-dimensional model coordinates.

The following transform models a standard camera with the lens pointing in a direction normal to the center of the image plane. The variable f specifies the distance of the image plane from the projection point, and usually does not need to be determined from the image when we are using a known camera (for convenience, we can let f represent the ratio of image distance to the width of the image plane, which means that image coordinates vary from 0 to 1 across the image). We must also specify a vector T giving the location of the camera lens in terms of world coordinates, and a rotation matrix R which depends on the camera orientation and maps points in world coordinates into points in a coordinate system with x and y axes parallel to the x and y axes of the camera film plane. Then the transform

$$(x, y, z) = R(p - T)$$

$$(x', y') = \left(\frac{fx}{z}, \frac{fy}{z}\right)$$

first transforms the point p in world coordinates into the point (x, y, z) in camera-based coordinates, and then creates the perspective projection of this point onto the image plane, with image coordinates (x', y'). Since this transform maps lines in the model into lines in the image, it is only necessary to transform the endpoints of a line in order to create its complete projection.

The most difficult aspect of the transformation is representing and working with the rotation R. Most work in computer graphics chooses to represent rotations with three-by-three matrices, but this representation is not very good for our purposes since it uses nine variables to represent something which has only three underlying parameters. Another possibility is to represent the rotation by giving its axis of rotation plus the angle of rotation about this axis. In fact, we can let the magnitude of the axis vector represent the magnitude of the rotation, and we have thus reduced the rotation to the minimal three parameters. However, the axis-angle representation requires a good deal of computation when we actually wish to rotate a point, and also makes it difficult to compose rotations. Quaternions [7] are a representation which combine the advantages of these other methods and have proved to be the most useful for our work. They use four variables to represent a rotation in such a way

that composition, normalization, rotation, and creation of a rotation about an arbitrary axis are all computationally efficient. However, the back-projection solution we are about to present is independent of any particular representation for rotations.

3. Back-projection using the Newton-Raphson method.

There are seven underlying parameters in the camera transform presented above—three parameters give the camera position T, three more are sufficient to specify the rotation R, and f specifies a property of the camera itself. Therefore, the back-projection problem is simply to calculate the values for these parameters which produce the best fit between an image and the projection of a model. However, because of problems in calculating a rotation in terms of its three underlying parameters, there appears to be no straightforward symbolic solution to the problem, and we are forced to seek an iterative solution. The method we have chosen is Newton-Raphson convergence, which has a fast quadratic convergence and can be cleanly applied to this problem. The Newton-Raphson method works by measuring each error in the current approximation and calculating the derivative of each of these error measurements with respect to each of the underlying parameters. We then create a system of linear equations which expresses each error as the sums of the correction to be made in each parameter times the derivative of the error with respect to that parameter. By solving this system of linear equations, we determine corrections to be added to the parameters which will correct for the originally measured errors. This technique works best when the derivatives are all fairly independent of one another, and are smooth enough over the error range for good convergence.

Unfortunately, the specification of the camera transform given in the previous section does not have simple derivatives of x' and y' with respect to the camera transform parameters. Once again, this is a result of the fact that it is difficult to represent a rotation in terms of its three underlying parameters. We have solved this difficulty by reparameterizing the camera transform to express it in terms of parameters that are related to the camera coordinate system rather than world coordinates. This new transform must be chosen carefully from among the many possibilities in order to keep the parameters as independent as possible from each other and to keep the derivatives simple. As before, our new transform specifies how a three-dimensional point p is to be mapped onto a point in the image (x', y'):

$$(x, y, z) = R(p)$$

$$(x', y') = \left(\frac{fx}{z + D_z} + D_x,\ \frac{fy}{z + D_z} + D_y\right)$$
$$= (fxc + D_x,\ fyc + D_y) \quad \text{where } c = \frac{1}{z + D_z}$$

Here the variables R and f remain the same as in the previous transform, but the vector T has been replaced by (D_x, D_y, D_z), where the two transforms are equivalent when

$$T = R^{-1}\left(-\frac{D_x(z + D_z)}{f},\ -\frac{D_y(z + D_z)}{f},\ -D_z\right)$$

The new parameterization is much better for our purposes, since D_x and D_y simply specify the location of the object on the image plane and D_z specifies the distance of the object from the camera; compare this with the very indirect specification of these same camera-related variables given by T. However, we have still solved only half the problem, since the three parameters underlying the rotation matrix are still difficult to express in a form closely related to the image. Our solution to this second problem was not to try to somehow express R in terms of image-centered parameters, but to take the initial specification of R as given and add to it incremental rotations ϕ_x, ϕ_y and ϕ_z about the x, y and z axes of the *camera* coordinate system. It is easy to compose rotations (especially when the quaternion representation of rotations is used as mentioned above), and the incremental rotations are fairly independent of one another if they are small. The Newton-Raphson method is now carried out by correcting errors in x' and y' by calculating the optimum correction rotations ϕ_x, ϕ_y and ϕ_z to be made about the image axes. Instead of adding these corrections to underlying parameters of R we create rotations of the given magnitudes about their respective coordinate axes and compose these new rotations with R.

One big advantage of using the ϕ's as our convergence parameters is that the derivatives of x, y, and z (and therefore of x' and y') with respect to them can be expressed in a strikingly simple form. For example, the derivative of x at a point (x, y) with respect to a counter-clockwise rotation of ϕ_z about the z axis is simply $-y$, since $(x, y) = (d\cos\phi_z,\ d\sin\phi_z)$ and therefore $\partial x/\partial\phi_z = -d\sin\phi_z = -y$. The following table gives these derivatives for all combinations of values:

	x	y	z
ϕ_x	0	$-z$	y
ϕ_y	z	0	$-x$
ϕ_z	$-y$	x	0

Partial derivatives of x, y and z with respect to counterclockwise rotations ϕ's (in radians) about the coordinate axes.

Given these derivatives it is straightforward to accomplish our original objective of calculating the partial derivatives of x' and y' with respect to each of the original camera parameters. For example, our transform tells us that:

$$x' = \frac{fx}{z + D_z} + D_x$$

so

$$\frac{\partial x'}{\partial D_x} = 1$$

and

$$\frac{\partial x'}{\partial \phi_y} = \frac{f}{z + D_z}\frac{\partial x}{\partial \phi_y} - \frac{fx}{(z + D_z)^2}\frac{\partial z}{\partial \phi_y}$$
$$= fcz + fc^2x^2 = fc(z + cx^2)$$

and

$$\frac{\partial x'}{\partial \phi_z} = \frac{f}{z + D_z}\frac{\partial x}{\partial \phi_z} = -fcy.$$

All the other derivatives can be calculated in a similar way, and this next table gives the derivatives of x' and y' with respect to each of the seven parameters of our camera model:

	x'	y'
D_x	1	0
D_y	0	1
D_z	$-fc^2x$	$-fc^2y$
ϕ_x	$-fc^2xy$	$-fc(z + cy^2)$
ϕ_y	$fc(z + cx^2)$	fc^2xy
ϕ_z	$-fcy$	fcx
f	cx	cy

Partial derivatives of x' and y' with respect to each of the camera transform parameters.

Given these partial derivatives of x' and y', it is easy to perform the convergence. For each point in the model which should match against some corresponding point in the image, we first calculate the camera transform of the model point and measure the error in its x component when compared to the given image point. We then create an equation which expresses this error E as the sum of the products of its partial derivatives times the error correction values:

$$\frac{\partial x'}{\partial D_x}\Delta D_x + \frac{\partial x'}{\partial D_y}\Delta D_y + \frac{\partial x'}{\partial D_z}\Delta D_z + \frac{\partial x'}{\partial \phi_x}\Delta \phi_x$$
$$+ \frac{\partial x'}{\partial \phi_y}\Delta \phi_y + \frac{\partial x'}{\partial \phi_z}\Delta \phi_z = E$$

Using the same point we create a similar equation for its y component, so for each point correspondence we derive two equations. From three point correspondences we can derive six equations and produce a complete linear system which can be solved for all six camera model corrections (we are assuming that the camera parameter f is either given, or can be approximated by a large value). After adding these corrections to our starting camera model, we have completed one iteration of the convergence. After each iteration the Δ terms should shrink by about one order of magnitude, and no more than a few iterations should be needed even for high accuracy.

In most applications of this method we will be given more correspondences between model and image than are strictly necessary, and we will want to perform some kind of best fit. In this case the Gauss least-squares method can easily be applied. The matrix equation given above can be expressed as

$$[A][\Delta] = [E]$$

where $[A]$ is the derivative matrix, $[\Delta]$ is the vector of unknown corrections, and $[E]$ is the vector of error terms. When this system is overdetermined, we can perform a least-squares fit of the errors simply by solving

$$[A]^T[A][\Delta] = [A]^T[E]$$

where $[A]^T[A]$ is square and has the correct dimensions for the vector $[\Delta]$.

The convergence properties of this solution are such that there should be few problems in picking the initial parameter values from which to converge. As long as the rotation errors ϕ_x, ϕ_y and ϕ_z are not greater than about 90 degrees, almost any values can be chosen for the other parameters. Usually, the source of the hypothesized matches carries a rough implication of the orientation of the object—for example, the quasi-invariants mentioned in the introduction are only applicable over a certain range of object orientations, so we have some idea of the object's orientation just by knowing which set of quasi-invariants were used.

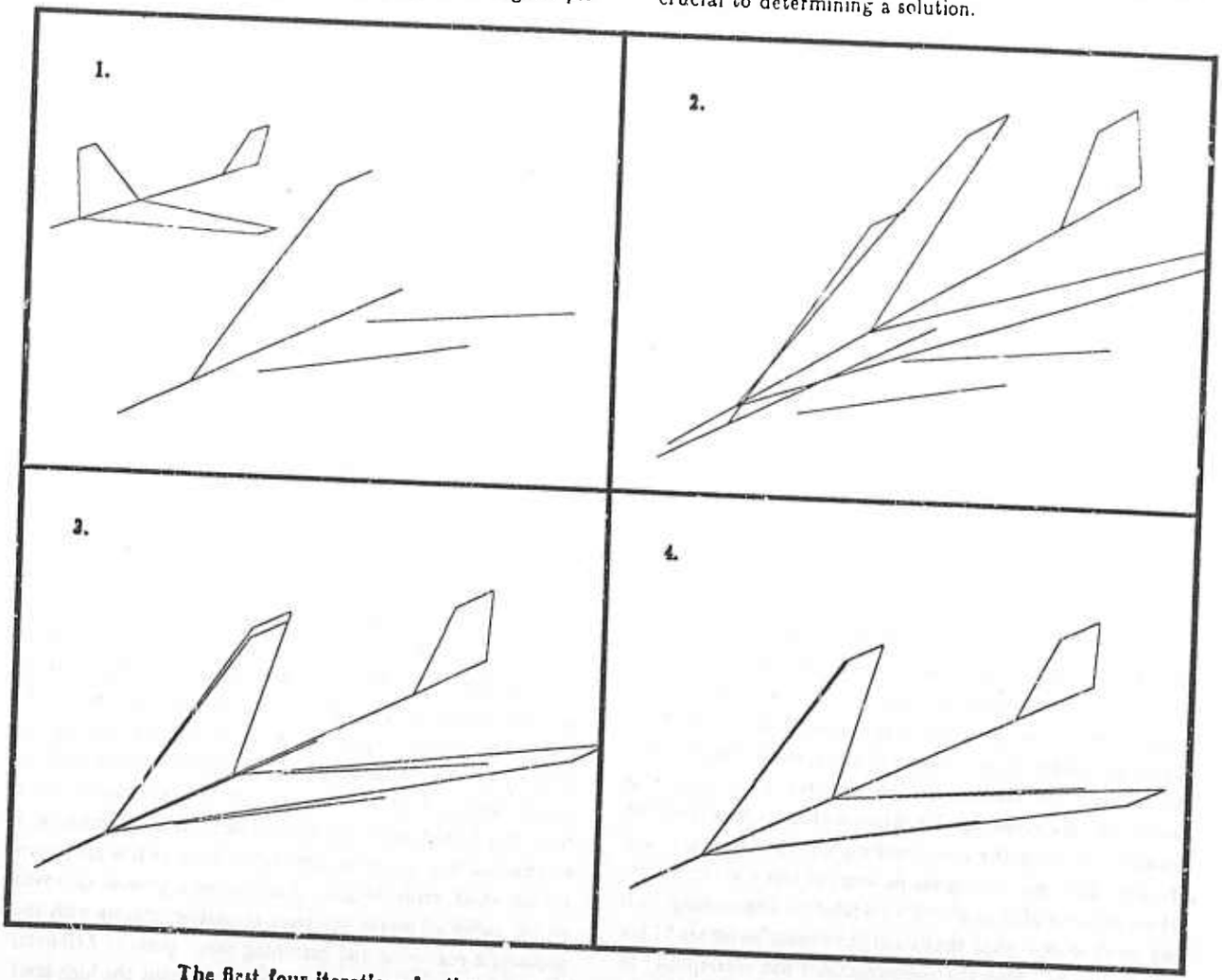
The situation may arise in which the values of some parameters are not deducible from the image and our solution is under-constrained. These situations can be handled by just not including the indeterminate parameters in our equations, but we have not yet developed a general method for deciding which parameters these are. We expect a solution to this problem shortly.

4. Using incompletely specified models

So far we have been describing the process of solving for the parameters which determine an object's position and orientation with respect to the external world. An important extension to this method is the ability to use models which

are parameterized internally, and have variable parts or articulations between parts. We can determine the values of these model parameters in the same way that we determine the correct projection parameters. The only requirement is that we be able to calculate the directional derivatives of points in the model with respect to the new parameters. For the common types of model parameterization, such as variable lengths or variable rotations about some axis, these derivatives are easily determined. From these it is easy to calculate the derivatives of projected image points and use them in the same way for convergence as before. We will now require more given correspondences between image and model in order to have a fully-determined system, but since each additional correspondence between an image point and a model point allows us to solve for two more unknown variables there should be little difficulty in meeting this requirement.

The power of this method can be best illustrated by giving an example. Assume that we want to recognize pictures of different types of airplanes, and we do not know in advance which type of airplane will be in a certain image. In this case our airplane model will have to be quite general and will not be able to give precise measurements for various lengths or such things as the angle between the wings and the fuselage. However, certain important constraints are known, such as the fact that the airplane will be symmetrical about the fuselage. This symmetry will be represented to the convergence algorithm by the fact that the model parameters referring to the right wing will be the same as those referring to the left wing, and any changes in these parameters refer to both wings. The convergence algorithm will then determine a camera transform and wing-fuselage angle which together produce the closest fit of model to image, as in the example below. Note that there may well be insufficient information to determine either the camera transform or the wing-fuselage angle independently, so the ability to solve for both simultaneously using knowledge of the plane's symmetry is crucial to determining a solution.



The first four iterations for the convergence of a simple model to fit some image lines.

Another type of constraint arises when we have some prior knowledge about the location of an object. For example, we may know the position of the ground plane relative to the camera and we can constrain the airplane to be positioned on the ground. In this case the airplane has only three degrees of freedom in its position (its x and y location on the ground and its orientation about the vertical axis). In this situation we do not need to solve for the full camera model, since this has already been determined relative to the ground plane. Instead, we can just solve for the parameters giving the position of the airplane relative to the ground using the techniques given above. This suggests that a more uniform description of the back-projection algorithm would be to treat the parameters which we have been calling "projection parameters" as just other kinds of model parameters which give the position of the entire object relative to camera space. This is not an entirely trivial change, as it will require a special mechanism to represent and work with the arbitrary rotation which can describe an object's orientation with respect to the camera. However, the change would make it conceptually easier to work with various situations in which we have multiple objects and only one camera. This is an area for further research.

One more extension that should be added to the basic algorithm is to allow the user to specify a legal range for each parameter. If the algorithm converges to a value for a parameter which is outside of its legal range, then this parameter can be set to its closest legal value and we can reconverge using the remaining parameters. This will produce a more accurate result when errors cause the least-squares fit to move a parameter to some value we know it cannot have.

5. Making use of line data

Another problem with the simple back-projection algorithm is that the initial image-model correspondences we derive are usually in terms of lines rather than points. This is because low-level vision routines are relatively good at finding the location of lines but are much less certain about exactly where the lines terminate. What we need to do is express our errors in terms of the distance of one line from another, rather than in terms of the error in the locations of points. The solution we have adopted is to measure as our errors the perpendicular distance of each endpoint of the model line from the corresponding line in the image, and to then take our derivatives in terms of this distance rather than in terms of x' or y'. This specifies mathematically what we want to say—that the model line should lie on top of the image line but that the endpoints need not correspond. In order to express the perpendicular distance of a point from a line it is useful to first express the line as an equation of the following form, in which m is the slope:

$$\frac{-m}{\sqrt{m^2+1}}x + \frac{1}{\sqrt{m^2+1}}y = d$$

In this equation d is the perpendicular distance of the line from the origin. If we substitute some point (x', y') into the left side of the equation and calculate the new value of d for this point (call it d'), then the perpendicular distance of this point from the line is simply $d - d'$. What is more, it is easy to calculate the derivatives of d' for use in the convergence, since the derivatives of d' are just a linear sum of the derivatives of x' and y' as given in the above equation, and we already know how to calculate the x' and y' derivatives from the solution given for using point correspondences. The result is that each line-line correspondence we are given between model and image gives us two equations for our linear system—the same amount of information that is conveyed by a point-point correspondence.

6. Implementation

The full back-projection method and extensions described above have been implemented as a module of the ACRONYM system. The algorithm has performed well and usually converges to the correct transform and parameter values to within about 1 part in 10^4 in less than 6 iterations. The algorithm is implemented in compiled MACLISP running on a DEC KL-10, and each iteration executes in less than 20 milliseconds.

The pictures on the previous page were produced by successive iterations towards a solution for fitting a simple airplane model to a few lines in a simulated image. The airplane model was parameterized so that the wings could be swept back and forth, and the algorithm was able to determine the airplane's orientation and the degree to which the wings were swept back from knowledge of the model's symmetry.

We have not yet fully integrated this process with the rest of the ACRONYM system, but it is fairly clear how most of this integration can be accomplished. Once a good approximation to the camera transform has been obtained, the model provides strong constraints on where to look for line segments corresponding to other parts of the model. A simple approach is to make use only of predictions which predict a line segment at a specific location in the image, and to examine the image for the instantiation of this prediction within small error bounds. A much more general approach would be to integrate the back-projection process with the geometric reasoning and matching components of Acronym, so that the constraints propagate throughout the high-level

predictions. An important aspect of this further processing would be techniques for discarding wild points in the the original matches, and reconverging to the best fit of the remaining matches. There should not be much difficulty in finally deciding whether a model matches at a certain location in the image, since the three-dimensional model makes so many specific predictions about the appearance of the image.

7. Conclusion.

The use of back-projection in conjunction with viewpoint independent models should enable a computer vision system to make effective use of the valuable constraints and predictions embodied in such models. It is interesting to note some apparent similarities between this use of models and the results of psychological experiments in human vision. The mental rotation of images and the role of expectations in perception are both active research areas in psychology. Shepard and Metzler [8] showed that the length of time subjects took to compare two pictures of a three-dimensional object was directly proportional to the three-dimensional angle of rotation between them. The classic work of Leeper [5] showed that people could often instantly identify degraded pictures when given an identifying label after being unable to make any sense of the picture when examining it without the label. In examples such as these, or in some cases where only simple line data is available, it appears that recognition is crucially dependent on expectations derived from some sort of prior visual representation, and little can be done if we are limited to information derived bottom-up from the image.

References

[1] Binford, Thomas O., "Visual Perception by Computer," Invited paper at *IEEE Systems Science and Cybernetics Conference*, Miami, Dec. 1971.

[2] Brooks, Rodney A. and Thomas O. Binford, "Representing and Reasoning about Partially Specified Scenes," *Proc. ARPA Image Understanding Workshop*, Baltimore, Apr. 1980.

[3] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "A Model-Based Vision System," *Proc. ARPA Image Understanding Workshop*, Cambridge, May 1978, 36-44.

[4] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "The ACRONYM Model-Based Vision System," *Proc. of IJCAI-79*, Tokyo, Aug. 1979, 105-113.

[5] Leeper R., "A study of a neglected portion of learning—the development of sensory organization." *Journal of Genetic Psychology*, 1935, 46:41-75.

[6] Rogers, David and J. Alan Adams, *Mathematical Elements for Computer Graphics*, McGraw-Hill, 1976.

[7] Salamin, Gene, "Application of Quaternions to Computation with Rotations" *Internal working paper, Stanford Artificial Intelligence Laboratory*, 1979.

[8] Shepard R. N. and J. Metzler, "Mental Rotation of Three-Dimensional Objects," *Science*, 1971, 171:701-703.

ASPECTS OF A COMPUTATIONAL THEORY OF HUMAN STEREO VISION

W.E.L. Grimson

MIT Artificial Inteligence Laboratory

545 Technology Square, Cambridge Ma 0213⁹ USA

ABSTRACT: Recently, Marr and Poggio (1979) presented a theory of human stereo vision. An implementation of that theory is presented, and consists of five steps: (1) The left and right images are each filtered with masks of four sizes that increase with eccentricity; the shape of these masks is given by $\nabla^2 G$, the laplacian of a gaussian function. (2) Zero-crossings in the filtered images are found along horizontal scan lines. (3) For each mask size, matching takes place between zero-crossings of the same sign and roughly the same orientation in the two images, for a range of disparities up to about the width of the mask's central region. Within this disparity range, Marr and Poggio showed that false targets pose only a simple problem. (4) The output of the wide masks can control vergence movements, thus causing small masks to come into correspondence. In this way, the matching process gradually moves from dealing with large disparities at a low resolution to dealing with small disparities at a high resolution. (5) When a correspondence is achieved, it is stored in a dynamic buffer, called the $2\frac{1}{2}$-dimensional sketch. To support the sufficiency of the Marr-Poggio model of human stereo vision, the implementation was tested on a wide range of stereograms from the human stereopsis literature. The performance of the implementation is illustrated and compared with human perception. As well, statistical assumptions made by Marr and Poggio are supported by comparison with statistics found in practice. Finally, the process of implementing the theory has led to the clarification and refinement of a number of details within the theory: these are discussed in detail.

1. Introduction

If two objects are separated in depth from a viewer, then the relative positions of their images will differ in the two eyes. This difference in relative positions -- the disparity — may be measured and used to estimate depth. The process of stereo vision, in essence, measures this disparity and uses it to compute depth information for surfaces in the scene.

The steps involved in measuring disparity are (Marr and Poggio, 1979): (S1) a particular location on a surface in the scene must be selected from one image; (S2) that same location must be identified in the other image; and (S3) the disparity between the two corresponding image points must be measured. The difficulty of the problem lies in steps (S1) and (S2), that is, in matching the images of the same location — the so-called correspondence problem. For the case of the human stereo system, it can be shown that this matching takes place very early in the analysis of an image, prior to any recognition of what is being viewed, using primitive descriptors of the scene. This is illustrated by the example of random dot patterns. Julesz (1960) demonstrated that two images, consisting of random dots when viewed monocularly, may be fused to form patterns separated in depth when viewed stereoscopically. Random dot stereograms are particularly interesting because when one tries to set up a correspondence between two arrays of dots, false targets occur in profusion. A *false target* refers to a possible but incorrect match between elements of the two views. In spite of such false targets, and in the absence of any monocular or high level cues, we are able to determine the correct correspondence. Thus, the computational problem of human stereopsis reduces to that of obtaining primitive descriptions of locations to be matched from the images, and of solving the correspondence problem for these descriptions.

A computational theory of the stereo process for the human visual system was recently proposed by Marr and Poggio (1979). According to this theory, the human visual processor solves the stereoscopic matching problem by means of an algorithm that consists of five main steps: (1) The left and right images are each filtered at different orientations with bar masks of four sizes that increase with eccentricity; these masks have a cross-section that is approximately the difference of two gaussian functions, with space constants in the ratio 1:1.75. Such masks essentially perform the operation of a second directional derivative after low pass filtering or smoothing, and can be used to detect changes in intensity at different scales. (2) Zero-crossings in the filtered images are found by scanning them along lines lying perpendicular to the orientation of the mask. Since convolving the image with the masks corresponds to performing a second directional derivative, the zero-crossings of the convolutions correspond to extrema in the first directional derivative of the image and thus to sharp changes in the original intensity function. (3) For each mask size, matching takes place between zero-crossing segments of the same sign and roughly the same orientation in the two images, for a range of disparities up to about the width of the mask's central region. Within this disparity range, Marr and Poggio showed that false targets pose only a simple problem, because of the roughly bandpass nature of the filters. (4) The output of the wide masks can control vergence movements, thus causing smaller masks to come into correspondence. In this way, the matching process gradually moves from dealing with large disparities at low resolution to dealing with small disparities at high resolution. (5) When a correspondence is achieved, it is stored in a dynamic buffer, called the $2\frac{1}{2}$-dimensional sketch (Marr and Nishihara, 1978).

An important aspect in the development of any computational theory is the design and implementation of an explicit algorithm for that theory. There are several benefits from such an implementation. One concerns the act of implementation itself which forces one to make all

details of the theory explicit. This often uncovers previously overlooked difficulties, thereby guiding further refinement of the theory.

A second benefit concerns the performance of the implementation. Any proposed model of a system must be testable. In this case, by testing on pairs of stereo images, one can examine the performance of the implementation, and hence of the theory itself, provided, of course, that the implementation is an accurate representation of that theory. In this manner, the performance of the implementation can be compared with human performance. If the algorithm differs strongly from known human performance, its suitability as a biological model is quickly brought into question (c.f. the cooperative algorithm of Marr and Poggio (1976)).

This article describes an implementation of the Marr-Poggio stereo theory, written with particular emphasis on the matching process (Grimson and Marr, 1979). For details of the derivation and justification of the theory, see Marr and Poggio (1979).

The first part of this paper describes the overall design of the implementation. Several examples of the implementation's performance on different images are then discussed, including random dot stereograms from the human stereopsis literature such as with one image defocussed, noise introduced into part of the images' spectra, and so forth. It is shown that the implementation behaves in a manner similar to humans on these special cases. Thirdly, the theory makes some statistical assumptions; these are compared with the actual statistics found in practice. Finally, the results of running the program on some natural images are shown.

2. Design of the program

The implementation is divided into five modules, roughly corresponding to the five steps in the summary above. These modules, and the flow of information between them, are illustrated in Figure 1. Each of the components is described in turn.

2.1 Input

There are two aspects of the human stereo system, embedded in the Marr-Poggio theory, which must be made explicit in the input to the algorithm. The first is the position of the eyes with respect to the scene, as eye movements will be critical for obtaining fine disparity information. The second is the change in resolution of analysis of the image with increasing eccentricity.

To account for these effects, the algorithm maintains as its initial input a stereo pair of images, representing the entire scene visible to the viewer. This pair of images corresponds to the environment around the visual system, rather than some integral part of the system itself. To create this representation of the scene, natural images were digitized on an Optronix Photoscan System P1000. The sizes of these images are indicated in the legends. Grey-level resolution is 8 bits, providing 256 intensity levels. For the random dot patterns illustrated in this article, the images were constructed by computer, rather than digitized from a photograph.

For a given position of the eyes, relative to the scene, a representation of the images on the two retinas is extracted. The algorithm creates this retinal representation by obtaining a second, smaller pair of images from the images representing the whole scene. The mapping from the scene images into the retinal images accounts for the two factors inherent in the Marr-Poggio theory. First, different sections of the scenes will be mapped to the center (fovea) of the retinal images as the positions of the eyes are varied. Since the matching process will take place on the array representing the retinal images, it is important that the coordinate systems of those arrays coincide with the current positions of the eyes. Note that the portion of the scene image which is mapped into the retinal image may differ for the two eyes, depending on the relative positions of the two optical axes. In particular, there may be differences in vertical alignment as well as in horizontal alignment. Second, the Marr-Poggio theory also states that the resolution of the earlier stages of the algorithm — the convolution and zero-crossings — scales linearly with eccentricity. The most convenient method for dealing with this fact is to account for the scaling with eccentricity at the level of the extraction of the images. This means that rather than extracting a set of retinal images in a linear manner, we may map the scene into the retinal images by a mapping whose magnification varies with eccentricity. By so doing, the later stages of processing need not explicitly account for the variation with eccentricity. Rather, these processes are considered as operating on a uniform grid. Note that this eccentric mapping is not essential, especially for small images. In most of the cases illustrated in this article, the mapping was not used.

After the completion of this stage, the implementation has created a representation of the images that has accounted for eye position and for retinal scaling with eccentricity. For each pass of the algorithm, the matching will take place on the representation of the retinal images, thereby implicitly assuming some particular eye positions. Once the matching has been completed, the disparity values obtained may be used to change the positions of the two optic axes, thus causing a new pair of retinal images to be extracted from the representations of the scene, and the matching process may proceed again.

2.2 Convolution

Given the retinal representations of the images, it is then necessary to transform them into a form upon which the matcher may operate. Marr and Poggio (1979) argued that the items to be matched in an image must be in one-to-one correspondence with well-defined locations on a physical surface. This led to the use of image predicates which correspond to changes in intensity. Since these intensity changes can occur over a wide range of scales within a natural image, they are detected separately at different scales. This is in agreement with the findings of Campbell and Robson (1968), who showed that visual information is processed in parallel by a number of independent spatial-frequency-tuned channels, and with the findings of Julesz and Miller (1975) and Mayhew and Frisby (1976), who showed that spatial-frequency-tuned channels are used in stereopsis and are independent. Recent work by Wilson and Bergen (1979) and Wilson and Giese (1977) provided evidence for the particular form of these spatial-frequency-tuned operators. Measuring contrast sensitivity to vertical line stimuli, Wilson and his collaborators showed that the image is convolved with an operator which in one dimension may be closely approximated by a difference of two gaussian functions (DOG).

In the original theory (Marr and Poggio, 1979), the proposed masks were oriented bar masks whose cross-section was a difference of two gaussians, as given by the Wilson and Bergen data. If an intensity change occurs along a particular orientation in the image, there

will be a peak in the first directional derivative of intensity, and a zero-crossing in the second directional derivative. Thus, the intensity changes in the image can be located by finding zero-crossings in the output of a second directional derivative operator. However, a number of practical considerations have led Marr and Hildreth (1979) to suggest that the initial operators not be directional operators. The only non-directional linear second derivative operator is the Laplacian. Marr and Hildreth have shown that provided two simple conditions on the intensity function in the neighbourhood of an edge are satisfied, the zero-crossings of the second directional derivative taken perpendicular to an edge will coincide with the zero-crossings of the Laplacian along that edge. Therefore, theoretically, we can detect intensity changes occuring at all orientations using the single non-oriented Laplacian operator. Thus, Marr and Hildreth propose that intensity changes occuring at a particular scale may be detected by locating the zero-crossings in the output of $\nabla^2 G$, the Laplacian of a gaussian distribution. The operator, together with its fourier transform, is illustrated in Figure 2. The form of the operator is given by:

$$\nabla^2 G(r,\theta) = \left[2 - \frac{r^2}{\sigma^2}\right] exp\{\frac{-r^2}{2\sigma^2}\}.$$

Given the form of the operators, it is only left to determine the size of these masks. To do this, we first note that Marr and Hildreth (1979) showed that the operator $\nabla^2 G$ is a close approximation to the DOG function. Wilson and Bergen's data indicated DOG filters whose sizes — specified by the width w of the filter's central excitatory region — range from 3.1' to 21' of visual arc. The variable w is related to the constant σ of $\nabla^2 G$ by the relation:

$$\sigma = \frac{w}{2\sqrt{2}}.$$

Wilson and Bergen's values were obtained by using oriented line stimuli. To obtain the diameter of the corresponding circularly symmetric center-surround receptive field, the values of w must be multiplied by $\sqrt{2}$. Finally, we want the resolution of the initial images to roughly represent the resolution of processing by the cones, and the size of the filters to represent the size of the retinal operators. In the densely packed region of the human fovea, the center-to-center spacing of the cones is 2.0 to 2.3 μm, corresponding to an angular spacing of 25 to 29 arc seconds (O'Brien, 1951). Accounting for the conversion of Wilson and Bergen's data, and using the figure of 27 seconds of arc for the separation of cones in the fovea, one arrives at values of w in the range 9 to 63 image elements, and hence, values of σ in the range 3 to 23 image elements.

Recently, it has been proposed (Marr, Poggio and Hildreth, 1979) that a further, smaller channel may be present. This channel would have a central excitatory width of $w = 1.5$', roughly corresponding to 4 image elements.

The present implementation uses four filters, each of which is a radially symmetric difference of gaussians, with w values of 4, 9, 17 and 35 image elements. The coefficients of the filters were represented to a precision of 1 part in 2048. Coefficients of less than $\frac{1}{2048}$'th of the maximum value of the mask were set to zero. Thus, the truncation radius of the mask (the point at which all further mask values were treated as zero) was approximately $1.8w$, or equivalently, 0.68σ.

The actual convolutions were performed on a LISP machine constructed at the MIT Artificial Intelligence Laboratory, using additional hardware specially designed for the purpose (Knight, et al. 1979). Figures 3 and 4 illustrates some images and their convolutions with varions sized masks.

After the completion of this stage of the algorithm, one has four filtered copies of each of the images, each copy having been convolved with a different size mask.

2.3 Detection and description of zero-crossings

According to the Marr-Poggio theory, the elements that are matched between images are (i) zero-crossings whose orientations are not horizontal, and (ii) terminations. The exact definition and hence the detection of terminations is at present uncertain; as a consequence, only zero-crossings are used as input to the matcher.

Since, for the purpose of obtaining disparity information, we may ignore horizontally oriented segments, the detection of zero-crossings can be accomplished by scanning the convolved image horizontally for adjacent elements of opposite sign, or for three horizontally adjacent elements, the middle one of which is zero, the other two containing convolution values of opposite sign. This gives the position of zero-crossings to within an image element.

In addition to their location, we record the sign of the zero-crossings (whether convolution values change from positive to negative or negative to positive as we move from left to right) and a rough estimate of the local, two-dimensional orientation of pieces of the zero-crossing contour. In the present implementation, the orientation at a point on a zero-crossing segment is computed as the direction of the gradient of the convolution values across that segment, and recorded in increments of 30 degrees. Figures 5 and 6 illustrate zero-crossings obtained in this way from the convolutions of Figures 3 and 4. Positive zero-crossings are shown white, and negative crossings, black.

We compute this zero-crossing description for each image and for each size of mask.

2.4 Matching

The matcher implements the second of the matching algorithms described by Marr and Poggio (1979, p.315). For each size of filter, matching consists of 6 steps:

(1) Fix the eye positions.

(2) Locate a zero-crossing in one image.

(3) Divide the region about the corresponding point in the second image into three pools.

(4) Assign a match to the zero-crossing based on the potential matches within the pools.

(5) Disambiguate any ambiguous matches.

(6) Assign the disparity values to a buffer.

These steps may be repeated several times during the fusion of an image. Given a position for the optic axes, these matching steps are performed, with the results stored in a buffer. These results may be used to refine the eye positions, causing a new set of retinal images to be extracted from the scene, and the matching steps are performed again.

We now expand upon each of the six steps of the matching process. The first step consists of fixing the two eye positions. The alignment between the two zero-crossing descriptions, corresponding to the positions of the optical axes, is determined in two ways. The initial offsets of the descriptions are arbitrarily set to zero. Thereafter, the offsets of the two optical axes are determined by accessing the current disparity values for a region and using these values to adjust the vergence of the eyes. In this implementation, this is done by modifying the extraction of the retinal images from the images of the entire scene, accounting for the positions of the optical axes.

Once the eye positions have been fixed, and the retinal images extracted, the images are convolved with the DOG filters, and the zero-crossing descriptions are extracted from the convolved images. For a zero-crossing description corresponding to a particular mask size, the matching is performed by locating a zero-crossing and executing the following operation. Given the location of a zero-crossing in one image, a horizontal region about the same location in the other image is partitioned into three pools. These pools form the region to be searched for a possible matching zero-crossing and consist of two larger convergent and divergent regions, and a smaller one lying centrally between them. Together these pools span a disparity range equal to $2w$, where w is the width of the central excitatory region of the corresponding two-dimensional convolution mask.

The following criteria are used for matching zero-crossings in the left and right filtered images, for each pool:

(1) the zero-crossings must come from convolutions with the same size mask.

(2) the zero-crossings must have the same sign.

(3) the zero-crossing segments must have roughly the same orientation.

A match is assigned on the basis of the number of pools containing a matching zero-crossing. If exactly one zero-crossing of the appropriate sign and orientation (within 30 degrees) is found within a pool, the location of that crossing is transmitted to the matcher. If two candidate zero crossings are found within one pool (an unlikely event), the matcher is notified and no attempt is made to assign a match for the point in question. If the matcher finds a single crossing in only one of the three pools, that match is accepted, and the disparity associated with the match is recorded in a buffer. If two or three of the pools contain a candidate match, the algorithm records that information for future disambiguation.

Once all possible unambiguous matches have been identified, an attempt is made to disambiguate double or triple matches. This is done by scanning a neighbourhood about the point in question, and recording the disparity sign of the unambiguous matches within that neighbourhood. (Disparity sign refers to the sign of the pool from which the match comes: divergent, convergent or zero.) If the ambiguous point has a potential match of the same disparity sign as the dominant type within the neighbourhood, then that is chosen as the match (this is the "pulling" effect). Otherwise, the match at that point is left ambiguous.

There is the possibility that the region under consideration does not lie within the $\pm w$ disparity range handled by the matcher. This situation is detected and handled by the following operation. Consider the case in which the region does lie within the disparity range $\pm w$. Excluding the case of occluded points, every zero-crossing in the region will have at least one candidate match (the correct one) in the other filtered image. On the other hand, if the region lies beyond the disparity range $\pm w$, then the probability of a given zero-crossing having at least one candidate match will be less than 1. In fact, Marr and Poggio show that the probability of a zero-crossing having at least one candidate match in this case is roughly 0.7. We can perform the following operation in this case. For a given eye position, the matching algorithm is run for all the zero-crossings. Any crossing for which there is no match is marked as such. If the percentage of matched points in any region is less than a threshold of 0.7 then the region is declared to be out of range, and no disparity values are accepted for that region.

The overall effect of the matching process, as driven from the left image, is to assign disparity values to most of the zero-crossings obtained from the left image. An example of the output appears in Figure 7. In this array, a zero-crossing at position (x, y) with associated disparity d has been placed in a three-dimensional array with coordinate (x, y, d). For display purposes, the array is shown in the figures as viewed from a point some distance away. The heights in the figure correspond to the assigned disparities.

After completion of this stage of the implementation, we have obtained a disparity array for each mask size. The disparity values are located only along the zero-crossing contours obtained from that mask.

2.5 Vergence Control

The Marr-Poggio theory states that in order to obtain fine resolution disparity information, it is necessary that the smallest channels obtain a matching. Since the range of disparity over which a channel can obtain a match is directly proportional to the size of the channel, this means that the positions of the eyes must be assigned appropriately to ensure that the corresponding zero-crossing descriptions from the two images are within a matchable range. The disparity information required to bring the smallest channels into their matchable range is provided by the larger channels. That is, if a region of the image is declared to be out of range of fusion by the smaller channels, one can frequently obtain a rough disparity value for that region from the larger channels, and use this to verge the eyes. In this way, the smaller channels can be brought into a range of correspondence.

Thus, after the disparities from the different channels have been combined, there is a mechanism for controlling vergence movements of the eyes. This operates by searching for regions of the image which do not have disparity values for the smallest channel, but which do have disparity values for the larger channels. These large channel values are used to provide a refinement to the current eye positions, thereby bringing the smaller channels into range of correspondence. Two possible mechanisms for extracting the disparity value from a region of the image include using the peak value of a histogram of the disparities in that neighbourhood, or using a local average of the disparity values. In the current implementation, the search for such a region proceeds outwards from the fovea.

It should be noted here that although the use of disparity information from coarser channels to drive eye movements, allowing smaller channels to come into correspondence, is a necessary condi-

tion of the Marr-Poggio theory, it is not necessarily the only such condition. In other words, there may be other modules of the visual system which can initiate eye movements, and thereby affect the input to the matching component, by altering the retinal images presented to the matcher. An example of this would be the evidence of Kidd et al. (1979) concerning the ability of texture contours to facilitate stereopsis by initiating eye movements. However, such effects are somewhat orthogonal to the question of the sufficiency of the matching component of the Marr-Poggio theory, since they affect the input to the matcher, but not the actual performance of the matching algorithm itself.

2.6 The $2\frac{1}{2}$-Dimensional Sketch

Once the separate channels have performed their matching, the results are combined and stored in a buffer, called the $2\frac{1}{2}$-D sketch. There are several possible methods for accomplishing this. As far as the Marr-Poggio theory is concerned, the important point is that some type of storage of disparity information occurs. (Perhaps the strongest argument for this is the fact that up to 2 degrees of disparity can be held fused in the fovea.)

We shall outline two different possibilities for the combination of the different channels. The method currently used in the implementation will be described below. A more biologically feasible method will be outlined in the discussion.

One of the critical questions concerning the form of the $2\frac{1}{2}$-D sketch is whether it reflects the scene or the retinal images. For all the cases illustrated in this article, the sketch was constructed by directly relating the coordinates of the sketch to the coordinates of the images of the entire scene. That is, as disparity information was obtained, it was stored in a buffer at the position corresponding to the position in the original scene from which the underlying zero-crossing came. Since disparity information about the scene is extracted from several eye positions, in order to store this information into a buffer, explicit information about the positions of the eyes is required. It will be argued in the discussion that this is probably inappropriate as a model of the human system. However, for the purposes of demonstrating the effectiveness of the matching module, such a representation is sufficient.

The actual mechanism for storing the disparity values requires some combination of the disparity maps obtained for each of the channels. Currently, the sketch is updated, for each region of the image, by writing in the disparity values from the smallest channel which is within range of fusion. Vergence movements are possible in order to bring smaller channels into a range of matching for some region. Further, for those regions of the image for which none of the channels can find matches, modification of the eye positions over a scale larger than that of the vergence movements is possible. By this method, one can attempt to bring those regions of the image into a range of fusion.

There are several possibilities for the actual method of driving the vergence movements. Two of these were outlined in the previous section.

The final output of the algorithm consists of a representation of disparity values in the image, those disparities being restricted to positions in the image lying along zero-crossings segments.

2.7 Summary of the process

The complete algorithm, as currently implemented, uses four mask sizes. Initially, the two views of the scene are mapped into a pair of retinal images. These images are convolved with each mask. The zero-crossings and their orientation are computed, for each channel and each view. The initial alignments of the eyes determine the registration of the images. The matching of the descriptions from each channel is performed for this alignment. Any points with either ambiguous matchings or with no match are marked as such.

Next, the percentage of unmatched points is checked, for all square neighbourhoods of a particular size. This size is chosen so as to ensure that the measurement of the statistics of matching within that neighbourhood is statistically sound. Only the disparity points of those regions whose percentage of unmatched points is below a certain threshold, determined by the statistical analysis of Marr and Poggio (1979), are allowed to remain. All other points are removed. The values which are kept are stored into a buffer. At this stage, vergence movements may take place, using information from the larger channels to bring the smaller channels into a range where matching is possible. Further, if there are regions of the image which do not have disparity values at any level of channel, an eye movement may take place in an attempt to bring those portions of the image into a range where at least the largest mask can perform its matching.

Note that the matching process takes place independently for each of the four channels. Once the matching of each channel is complete, the results are combined into a single representation of the disparities.

The final output is thus a disparity map, with disparities assigned along most portions of the zero-crossing contours obtained from the smallest masks. The accuracy of the disparities thus obtained depends on how accurately the zero-crossings have been localized, which may, of course, be to a resolution much finer than the initial array of intensity values that constitutes the image.

3. Examples and Assessment of Performance

A standard tool in the examination of human stereo perception is the random dot stereogram (Julesz, 1960, 1971). This is a pair of stereo images where each image, when viewed monocularly, consists only of randomly distributed dots, yet when viewed stereoscopically, may be fused to yield patterns separated in depth. Such patterns are a useful tool for analysing the stereo component of the human visual system, since there are no visual cues other than the stereoscopic ones. We can test the sufficiency of the algorithm by comparing human perception with the performance of the algorithm on such patterns. As well, since random dot stereograms have well demarked disparity values, it is easy to assess the correctness of the algorithm's performance on such patterns.

Table 1 lists some of the matching statistics for various random dot patterns. These are illustrated in Figures 8–13 and discussed below.

The first pattern consisted of a central square separated in depth from a second plane. The pattern had a dot density of 50% and its analysis is shown in Figure 7. Each dot was a square with four image elements on a side. For the algorithm, this corresponds to a dot of approximately two minutes of visual arc. The total pattern was 320 image elements on a side. The central plane of the figure was shifted 12 image

elements in one image relative to the other. The final disparity map assigned after the matching of the smallest channel had the following statistics. The number of zero-crossing points in the left description which were assigned a disparity was 11847. Of these 11847, 11830 were disparity values which were exactly correct, and an additional 14 deviated by one image element from the correct value. Approximately 0.03% of the matched points, or roughly 3 points in 10000 were incorrectly matched.

A similar test was run on patterns with a dot density of 25%, 10% and 5%. The results are illustrated in Figure 8.

For each of these cases, the number of incorrectly matched points was extremely low. Those points which were assigned incorrect disparities all occured at the border between the two planes, that is, along the discontinuity in disparity.

A more complex random dot pattern consisted of a wedding cake, built from four different planar layers, each separated by 8 image elements, or 2 dot widths. This is illustrated in Figure 9.

In this case, the number of zero-crossing points assigned a disparity was 11162. Of these points, 11095 were assigned a disparity value which was exactly correct, and an additional 61 deviated from the correct value by one image element. Approximately 0.06% of the points were incorrectly matched. Again, these incorrect points all occured at the boundaries between the planes. A second complex pattern is illustrated in Figure 9. The object is a spiral with a range of continuously varying disparities.

There are a number of special cases of random dot patterns which have been used to test various aspects of the human visual system. The algorithm was also tested on several of these stereograms. They are outlined below and a comparison between the performance of the algorithm, and human perception is given.

It is known that if one or both of the images of a random dot stereogram are blurred, fusion of the stereogram is still possible (Julesz 1971, p.96). To test the algorithm in this case, the left half of a 50% density pattern was blurred by convolution with a gaussian mask. This is illustrated in Figure 10. The disparity values obtained in this case were not as exact as in the case of no blurring. Rather, there was a distribution of disparities about the known correct values. As a result, the percentage of points that might be considered incorrect (more than one image element deviation from the correct value) rose to 6%. However, the qualitative performance of the algorithm is still that of two planes separated in depth. It is interesting to note that slight distribution of disparity values about those corresponding to the original planes is consistent with the human perception of a pair of slightly warped planes.

Julesz and Miller (1975) showed that fusion is also possible in the presence of some types of masking noise. In particular, if the spectrum of the noise is disjoint from the spectrum of the pattern, it can be demonstrated that fusion of the pattern is still possible. Within the framework of the Marr-Poggio theory, this is equivalent to stating that if one introduces noise of such a spectrum as to interfere with one of the stereo channels, fusion is still possible among the other channels, provided the noise does not have a substantial spectral component overlapping other channels as well. This was tested on the algorithm by high pass filtering a second random dot pattern, to create the noise, and adding the noise to one image. In the case illustrated in Figures 10 and 11, the spectrum of the noise was designed to interfere maximally with the smallest channel. In the case shown by HNOISE1 and HNOISE2 in Table 1, the noise was added such that the maximum magnitude of the noise was equal to the maximum magnitude of the original image. HNOISE1 illustrates the performance of the smallest channel. HNOISE2 illustrates the performance of the next larger channel. It can be seen that for this case, some fusion is still possible in the smallest channel, although it is patchy. The next larger channel also obtains fusion. In both cases, the accuracy of the disparity values is reduced from the normal case. This is to be expected, since the introduction of noise tends to displace the positions of the zero-crossings. In the case shown by HNOISE3 and HNOISE4 in Table 1, the noise was added such that the maximum magnitude was twice that of the maximum magnitude of the original image. Here, matching in the smallest channel is almost completely eliminated (HNOISE3). Yet matching in the next larger channel is only marginally affected (HNOISE4).

The implementation was also tested on the case of adding low pass filtered noise to a random dot pattern, with results similar to that of adding high pass filtered noise. Here, the larger channels are unable to obtain a good matching, while the smaller channels are relatively unaffected.

If one of the images of a random dot pattern is compressed in the horizontal direction, the human stereo system is still able to achieve fusion (Julesz 1971, p.213). The algorithm was tested on this case, and the results are shown in Figure 11. It can be seen that the program still obtains a reasonably good match. The planes are now slightly slanted, which agrees with human perception.

If some of the dots of a pattern are decorrelated, it is still possible for a human observer to achieve some kind of fusion (Julesz 1971, p.88). Two different types of decorrelation were tested. In the first type, increasing percentages of the dots in the left image were decorrelated at random. In particular, the cases of 10%, 20% and 30% were tried, and are illustrated in Figure 12. For the 10% case, (table entry Uncorr1) it can be seen that the algorithm was still able to obtain a good matching of the two planes, although the total number of zero-crossings assigned a disparity decreased, and the percentage of incorrectly matched points increased. When the percentage of decorrelated dots was increased to 20% (table entry Uncorr2), the number of matched points decreased again, although the percentage of those which were incorrectly matched remained about the same. Finally, when the percentage of decorrelated dots was increased to 30% (table entry Uncorr3), the algorithm found virtually no section of the image which could be fused.

The failure of the algorithm to match the 30% decorrelated pattern is caused by the component of the algorithm which checks that each region of the image is within range of correspondence. Recall that in order to distinguish between the case of two images beyond range of fusion (for the current eye positions) which will have only randomly matching zero-crossings, and the case of two image within range of fusion, the Marr-Poggio theory requires that the percentage of unmatched points is less than some threshold. This threshold is approximately 0.3, according to the statistical analysis of Marr and Poggio (1979). For the case of the pattern with 30% decorrelation, on the average, each region of the image will have roughly 30% of its zero-crossings different and hence the algorithm decides that the region is out of range of correspondence. Hence, no disparities are accepted for this region.

For the algorithm, the computational reason for the failure to process patterns with 30% decorrelation is that it could not distinguish a correctly matched region of such a pattern from a region which was

out of range of correspondence, but had a random set of matches for many of the points in the region. It is interesting to note that many human subjects observe a similar behavior; that is, some kind of fusion for up to 20% decorrelation, although the fusion becomes increasingly weaker, and virtually no fusion for patterns with 30% decorrelation.

One can also decorrelate the pattern by breaking up all white triplets along one set of diagonals, and all black triplets along the other set of diagonals (Julesz 1971, p.87). The table entry Uncorrd indicates the matching statistics for this case. Again, it can be seen that the program still obtains a good match, as do human observers. The performance of the algorithm is illustrated in Figure 13.

4. Statistics

A number of parameters are important for the theory, which makes assumptions about them, and they have been measured on random dot images. The worst cases occur for patterns with a density of 50%, and for such patterns the worst case values encountered for the parameters have the values shown in Table 2. The theoretical worst case bounds used by Marr and Poggio appear for comparison.

5. Natural Images

One can test the implementation on natural scenes, as well as random dot stereograms. In this case, it is more difficult to assess the exact performance of the algorithm. However, qualitative performance can be measured and some examples are shown in Figure 14. In these cases, as well as other natural images on which the algorithm has been tested, the disparity values obtained by the algorithm are seen to be in good qualitative agreement with the shapes of the surfaces in the scenes.

6. Acknowledgements

Without David Marr and Tomaso Poggio, this work would have been impossible. Ellen Hildreth, Keith Nishihara and Shimon Ullman provided many useful comments and suggestions.

7. References

Braddick, O. 1978 Multiple matching in stereopsis. (unpublished MIT report).

Campbell, F.W. and Robson, J. 1968 Application of Fourier analysis to the visibility of gratings. *J. Physiol., Lond.* 197, 551-566.

Grimson, W.E.L. A refinement of a computational theory of human stereo vision *in preparation.*

Grimson, W.E.L. and Marr, D. 1979 A computer implementation of a theory of human stereo vision. *Proceedings: Image Understanding Workshop* 41-47.

Julesz, B. 1960 Binocular depth perception of computer-generated patterns. *Bell System Tech. J.* 39, 1125-1162.

Julesz, B. 1971 *Foundations of cyclopean perception.* Chicago: The University of Chicago Press.

Julesz, B. and Miller, J.E. 1975 Independent spatial-frequency-tuned channels in binocular fusion and rivalry. *Perception* 4 125-143.

Kidd, A.L., Frisby, J.P. and Mayhew, J.E.W. 1979 Texture contours can facilitate stereopsis by initiating appropriate vergence eye movements. *Nature* 280, 829-832.

Knight, T.F., Moon, D.A., Holloway, J., and Steele, G.L. 1979 CADR *MIT Artificial Intelligence Laboratory Memo* 528.

Marr, D. and Hildreth, E. 1980 Theory of edge detection. *Proc. R. Soc. Lond.* (in the press).

Marr, D. and Nishihara, H.K. 1978 Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. R. Soc. Lond.* B. 200, 269-294.

Marr, D. and Poggio, T. 1976 Cooperative computation of stereo disparity. *Science, N.Y.* 194, 283-287.

Marr, D. and Poggio, T. 1979 A computational theory of human stereo vision. *Proc. R. Soc. Lond.* B. 204, 301-328.

Marr, D., Poggio, T. and Hildreth, E. 1979 The smallest channel in early human vision. *JOSA* (submitted for publication).

Mayhew, J.E.W. and Frisby, J.P. 1976 Rivalrous texture stereograms *Nature, Lond.* 264, 53-56.

O'Brien, B. 1951 Vision and resolution in the central retina. *J. Opt. Soc. Am.* 41, 882-894.

Ullman, S. 1979 *The interpretation of visual motion* Cambridge: MIT Press.

Wilson, H.R. and Bergen, J.R. 1979 A four mechanism model for spatial vision. *Vision Res.* (in the press).

Wilson, H.R. and Giese, S.C. 1977 Threshold visibility of frequency gradient patterns. *Vision Res.* 17, 1177-1190.

TABLE OF MATCHES						
pattern	density	total	exact	one pixel	wrong	%wrong
square	50%	11847	11830	14	3	.03
square	25%	9661	9632	22	7	.07
square	10%	5286	5264	20	2	.04
square	5%	3500	3498	0	2	.06
wedding	50%	11162	11095	61	6	.06
hnoise1	50%	2270	1909	346	15	.7
hnoise2	50%	8683	6621	1868	194	2.
hnoise3	50%	63	28	24	11	17.
hnoise4	50%	8543	5194	2864	485	6.
uncorr1	50%	9545	9091	263	191	2.
uncorr2	50%	4343	4120	143	80	2.
uncorr3	50%	134	127	2	5	4.
uncorrd	50%	6753	6325	271	157	2.

Table 1.

TABLE OF STATISTICS				
parameter	expected worst case behavior	large channel w = 35	medium channel w = 17	small channel w = 9
average distance between zero-crossings of same sign	2 w	1.51 w	1.88 w	1.87 w
probability of candidates in at most one pool	> .50	.77	.75	.69
probability of candidates in two pools	< .45	.21	.25	.31
probability of candidates in all three pools	< .05	.02	.01	.01
given a candidate near zero, probability of no other candidates	> .9	.88	.85	.87

Table 2.

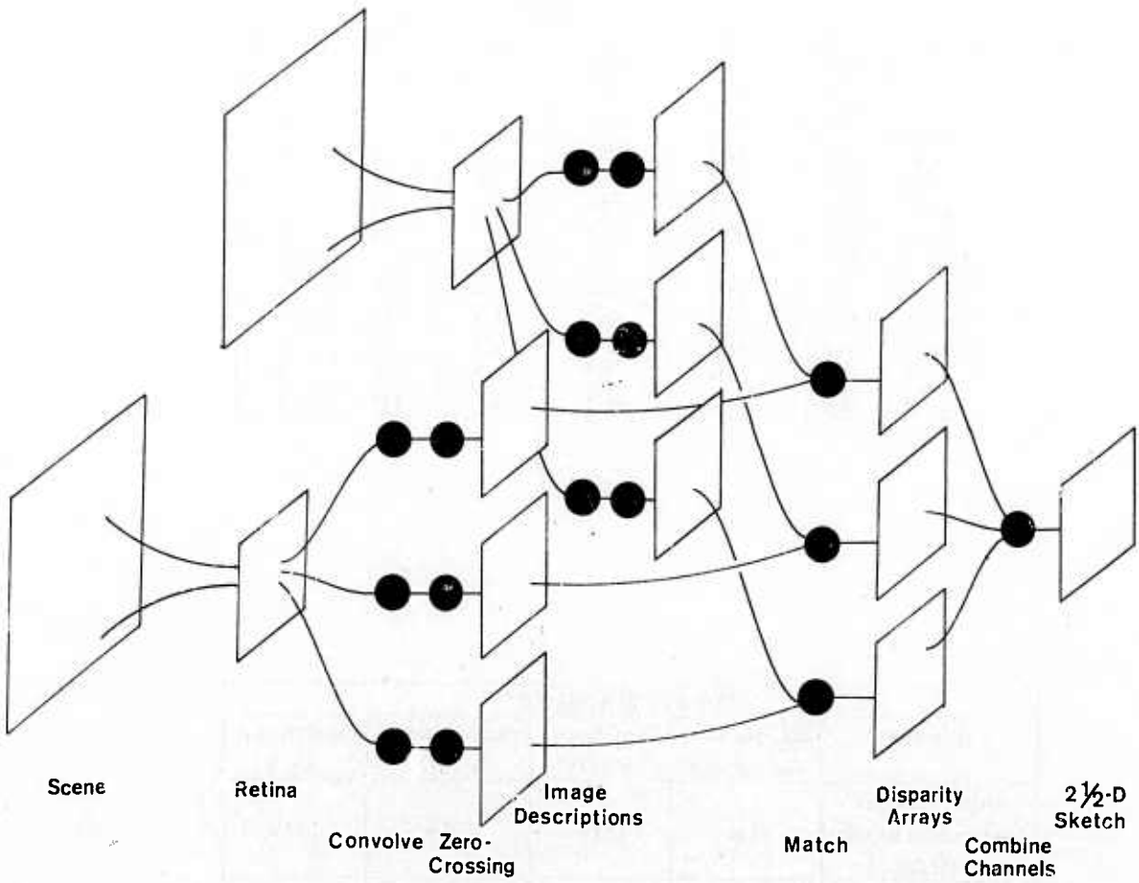


Figure 1. Diagram of the algorithm. The images of the scene are mapped into the images of the retinas, taking into account the eye positions. Each image is convolved with a set of different sized masks and zero-crossings are located for each convolution. For each size mask, the left and right zero-crossing descriptions are matched. These are combined into a single representation. As well, the matches from the larger channels can drive eye vergence movements, causing new retinal images to be created and allowing the smaller channels to come into correspondence.

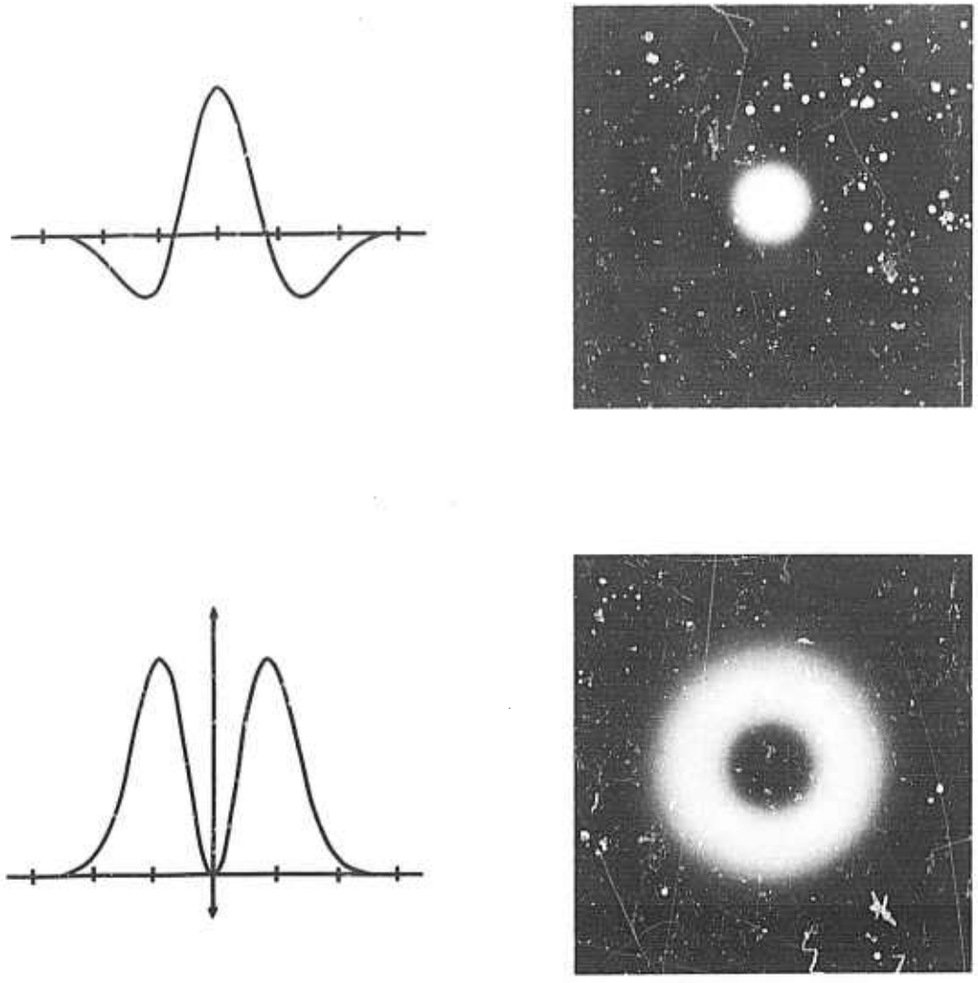


Figure 2. The operators G'' and $\nabla^2 G$. The top left figure show G'', the second derivative of a one-dimensional guassian distribution. The top right figure shows $\nabla^2 G$, its rotationally symmetric two-dimensional counterpart. The bottom figures show their Fourier transforms.

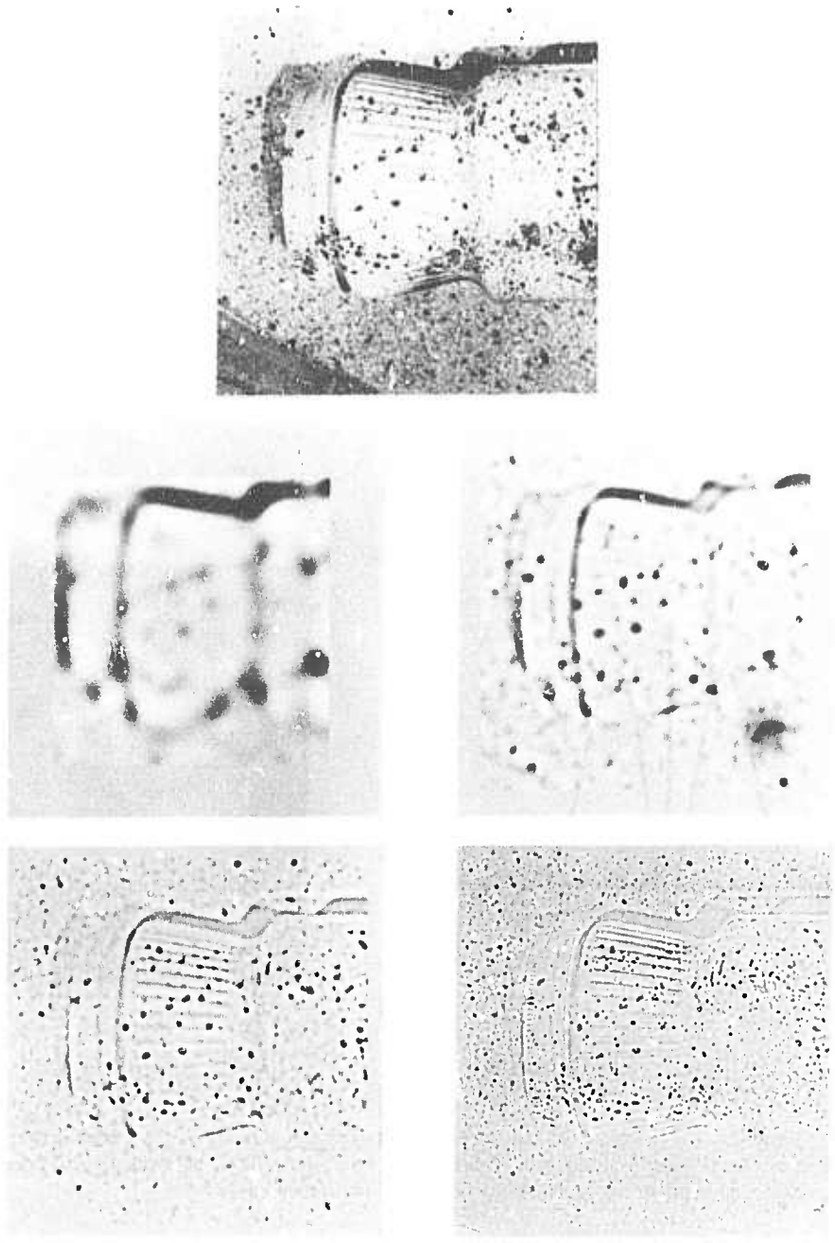


Figure 3. Examples of convolutions with $\nabla^2 G$. The top figure shows a natural image. The bottom figures show the convolution of this image with a set of $\nabla^2 G$ operators. The sizes of these operators are $w = 36, 18, 9$ and 4 image elements.

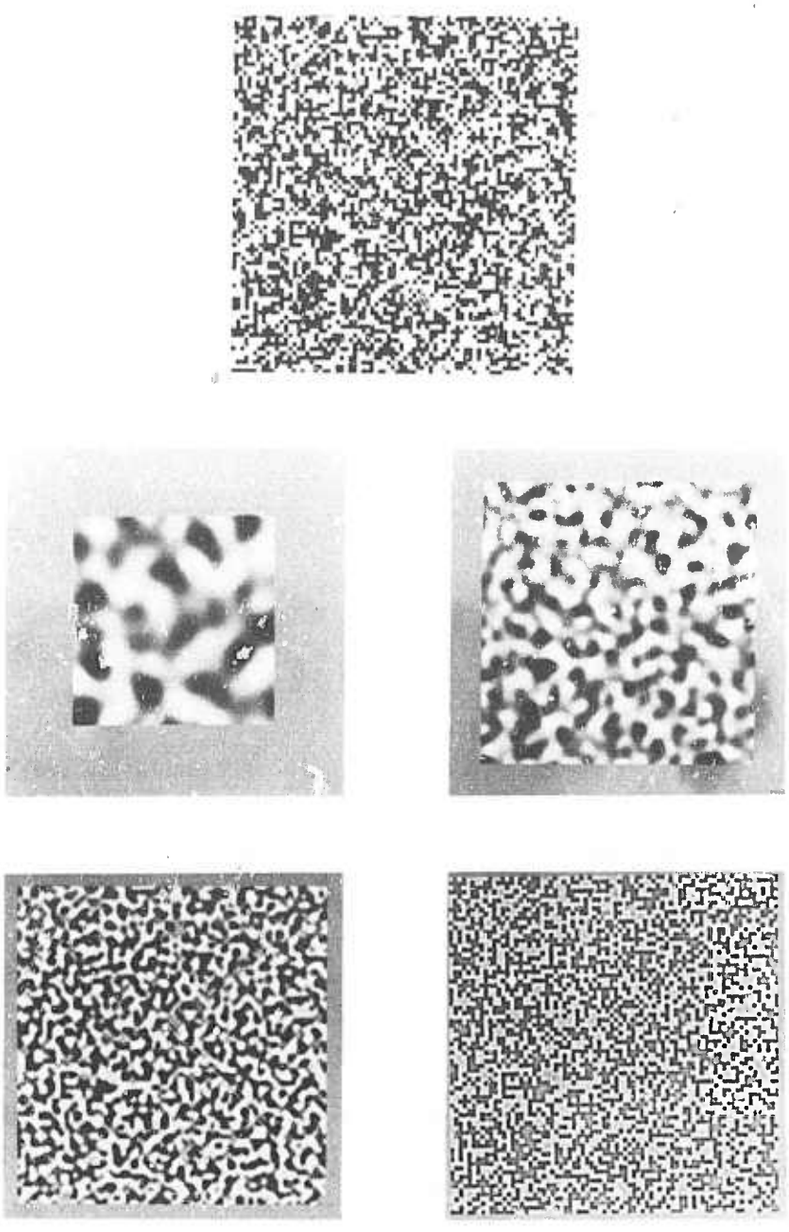


Figure 4. Examples of convolutions with $\nabla^2 G$. The top figure shows a random dot pattern. The bottom figures show the convolution of this image with a set of $\nabla^2 G$ operators. The sizes of these operators are $w = 36, 18, 9$ and 4 image elements.

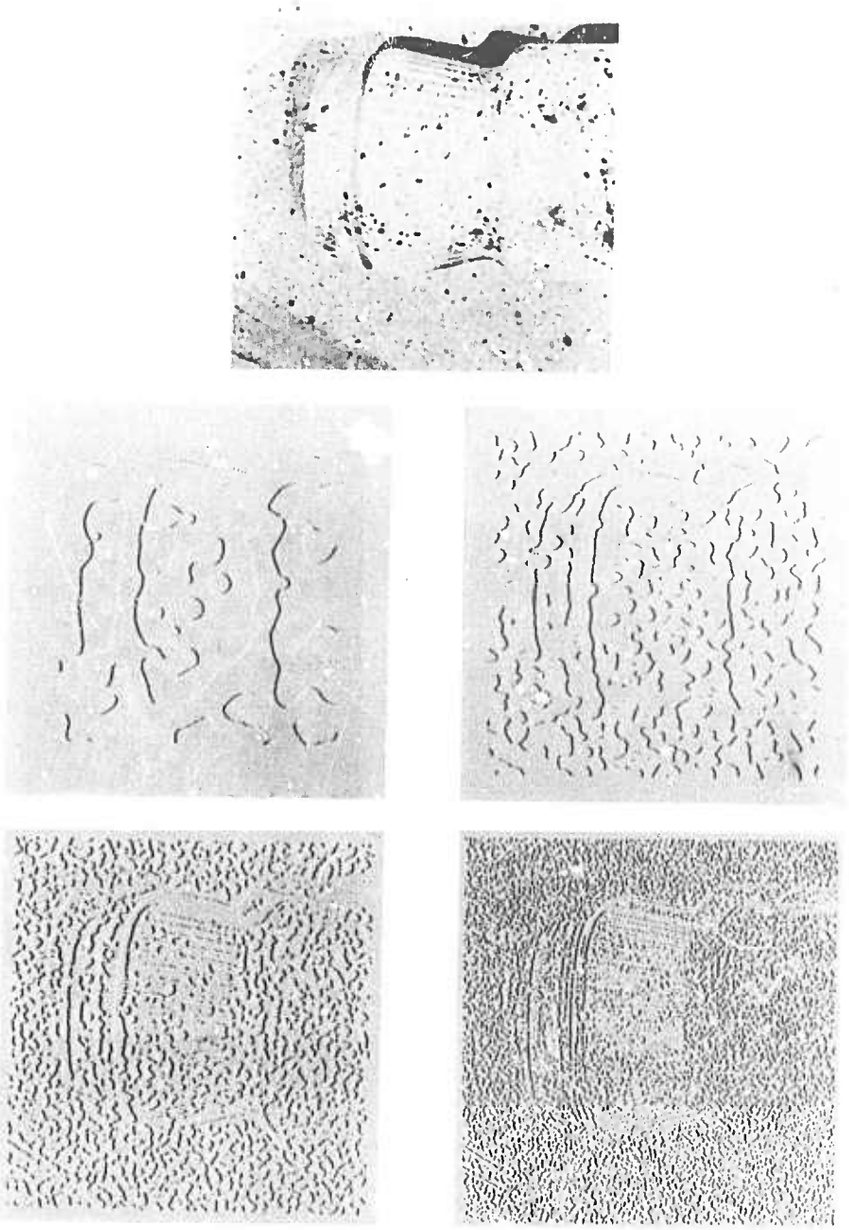


Figure 5. Examples of zero-crossing descriptions. The top figure show a natural image. The bottom figures show the zero-crossings obtained from the convolutions of Figure 3. The white lines mark positive zero-crossings and the black lines, negative ones.

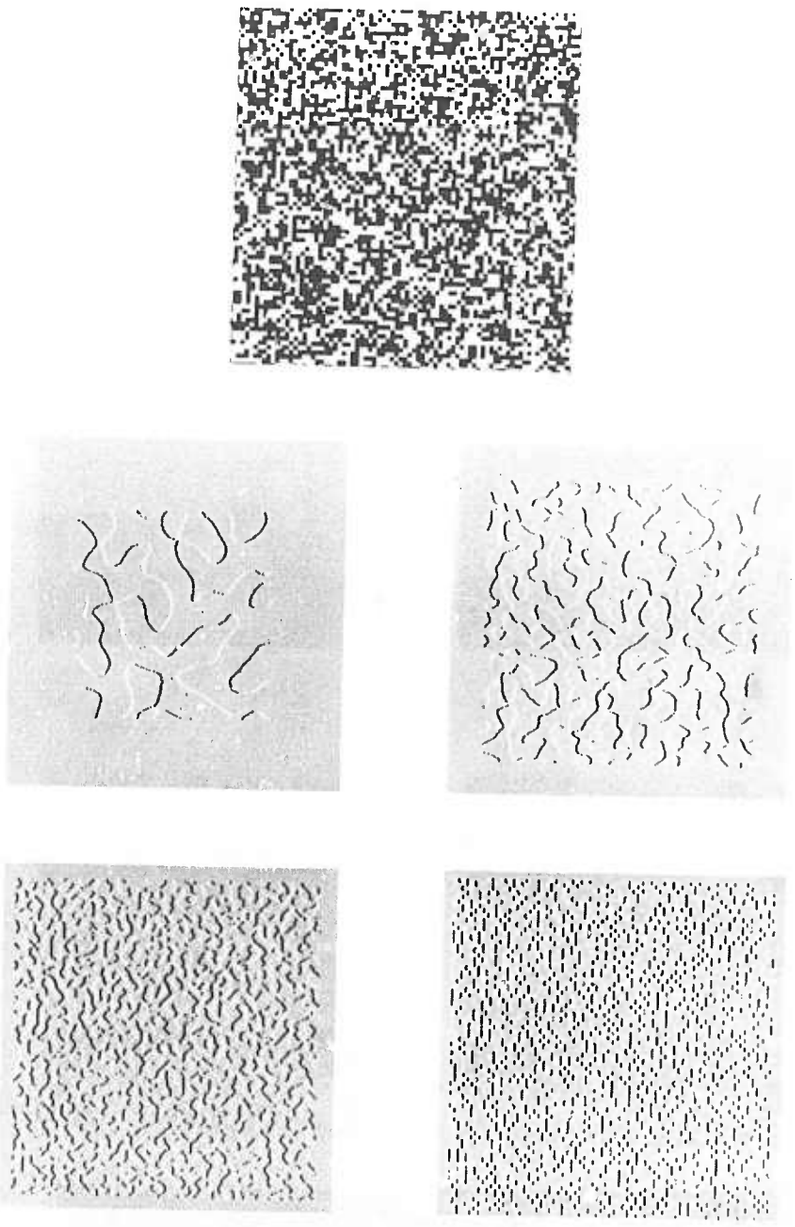


Figure 6. Examples of zero-crossing descriptions. The top figure show a random dot pattern. The bottom figures show the zero-crossings obtained from the convolutions of Figure 4. The white lines mark positive zero-crossings and the black lines, negative ones.

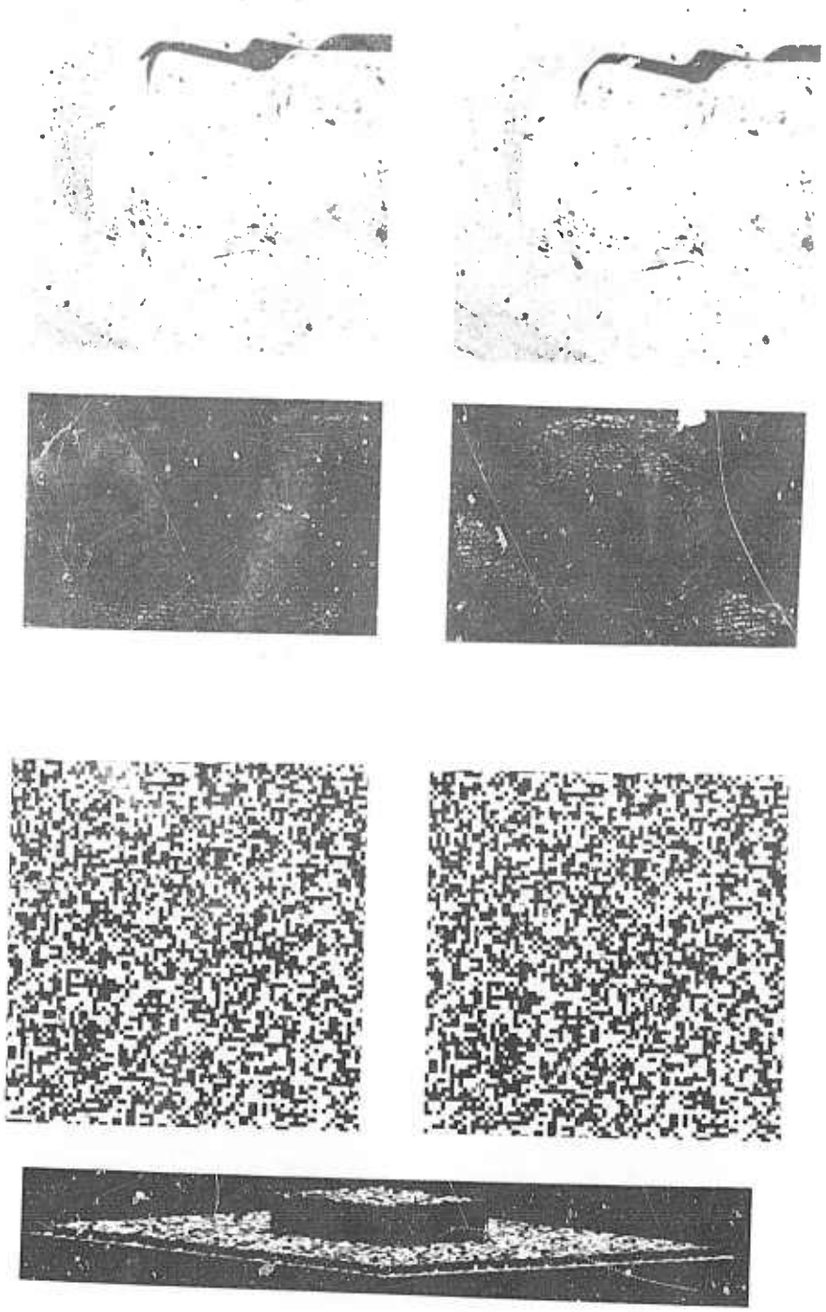


Figure 7. Results of the algorithm. The top stereo pair is an image of a painted coffee jar. The next two figures show two orthographic views of the disparity map. The disparities are displayed as $\{x, y, c - ad(x, y)\}$, where c is a constant and $d(x, y)$ is the difference in the location of a zero-crossing in the right and left images. For purposes of illustration, a has been adjusted to enhance the features of the disparity map. The left view of the disparity map shows the jar as viewed from the lower edge of the image, and the right view show the jar as viewed from the left edge of the image. Note that the background plane appears tilted in the disparity map. This agrees with the fused perception. The second stereo pair is a 50% density random dot pattern. The bottom figure shows the disparity map as viewed orthographically from some distance away. All disparity maps are those obtained from the $w = 4$ channel.

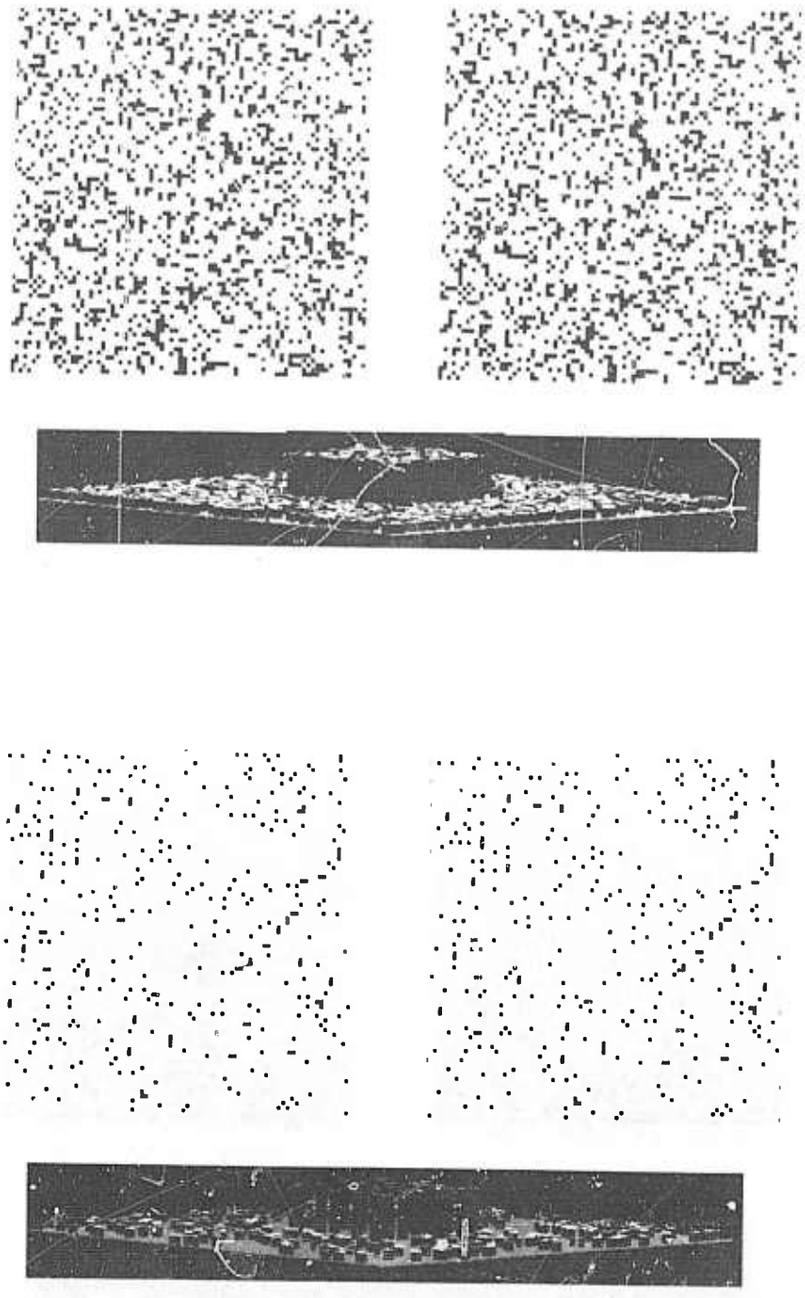


Figure 8. The top stereo pair is a 25% density random dot pattern. The disparity map below it is displayed as in Figure 7. The bottom stereo pair is a 5% density random dot pattern. Its disparity map is shown below it. Both disparity maps are obtained from the $w = 4$ channel.

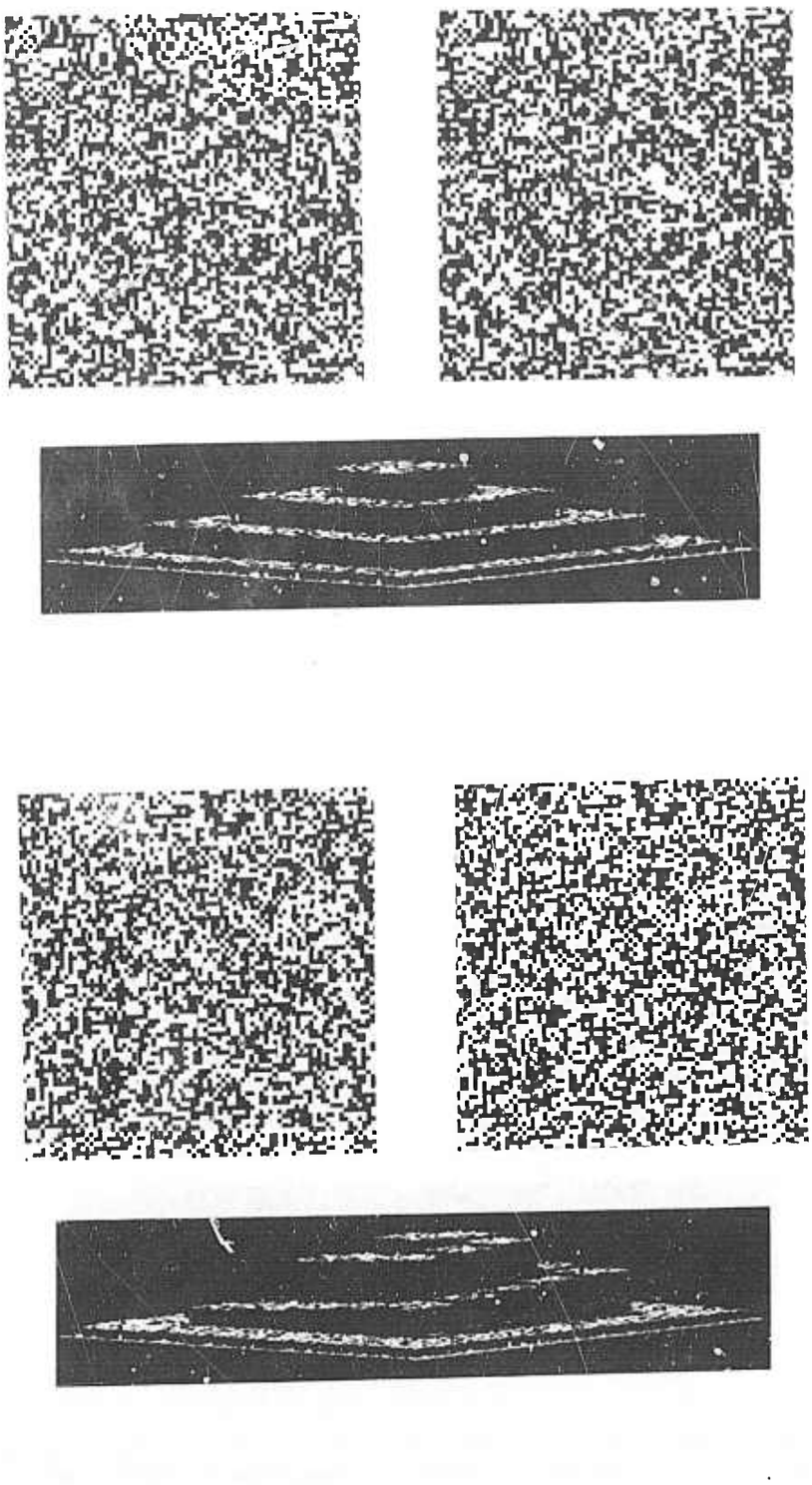
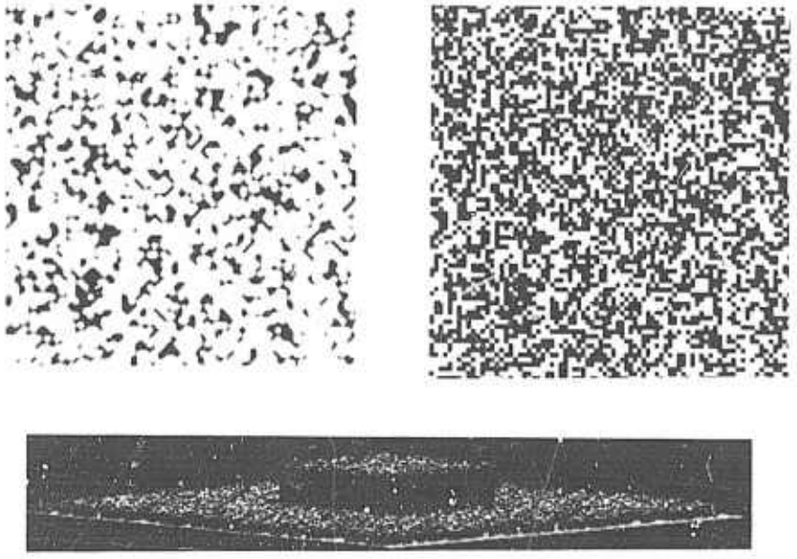


Figure 9. The top stereo pair is a 50% density wedding cake, composed of four planar levels. The disparity map is shown below it. The bottom stereo pair is a 50% spiral. The disparity map is shown below it, in a manner similar to Figure 7. Both disparity maps are obtained from the $w = 4$ channel.



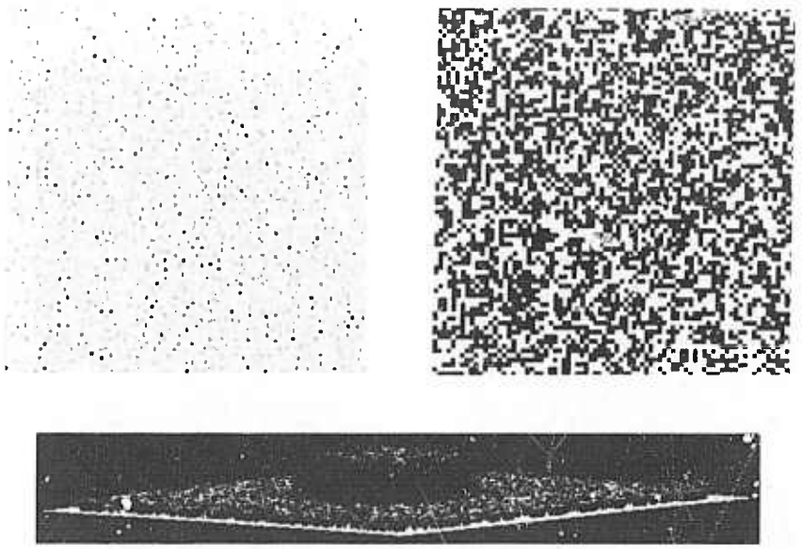


Figure 10. The top stereo pair is a 50% density pattern in which the left image has been blurred. The disparity map is shown below it. It can be seen that two planes are still evident, although they are not as sharply defined as in Figure 7 or Figure 8. The disparity map is that obtained from the $w = 4$ channel. The bottom stereo pair is a 50% density pattern. The left image has had high pass filtered noise added to it so that the maximum magnitude of the noise is equal to the maximum magnitude of the image. The disparity map shown is that obtained by the $w = 9$ channel.

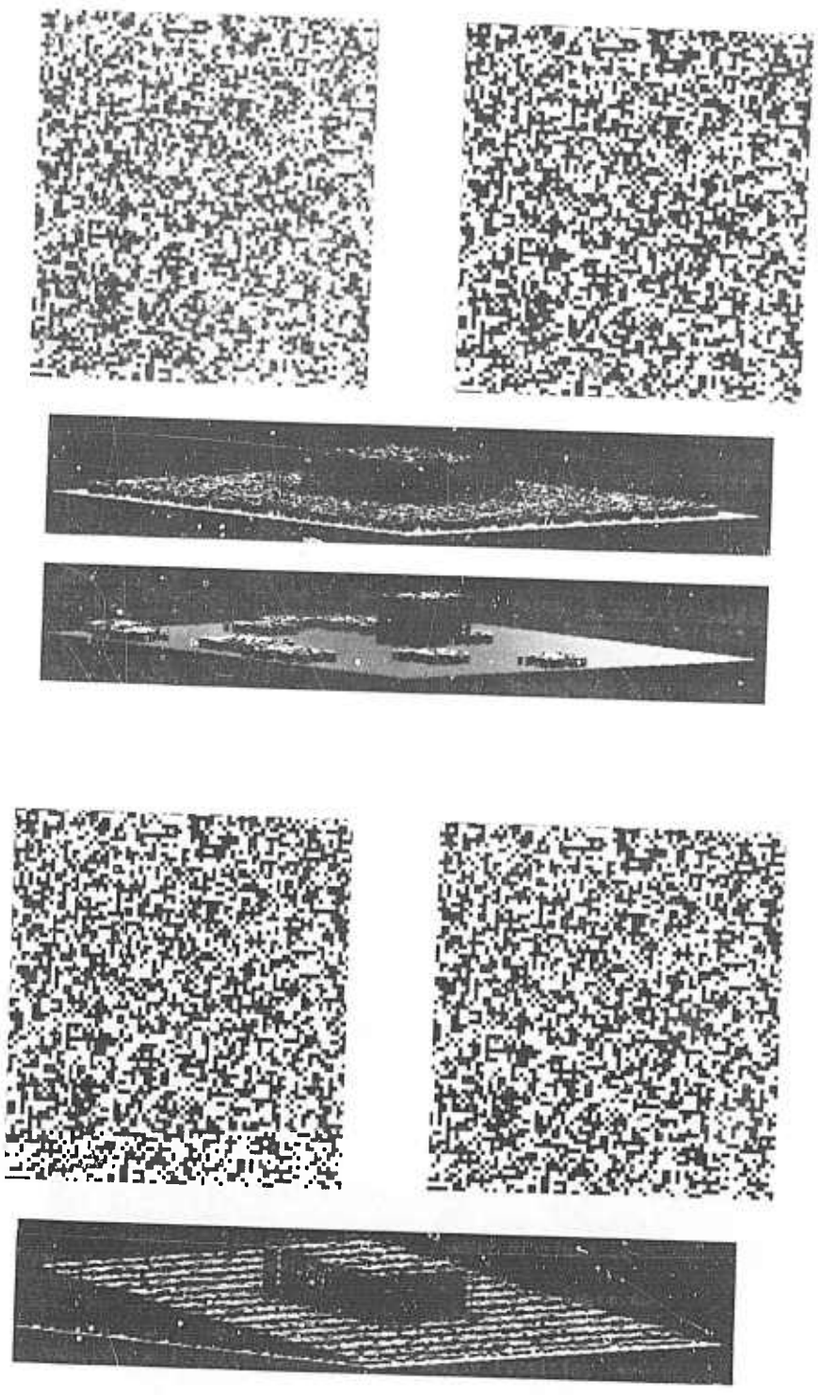
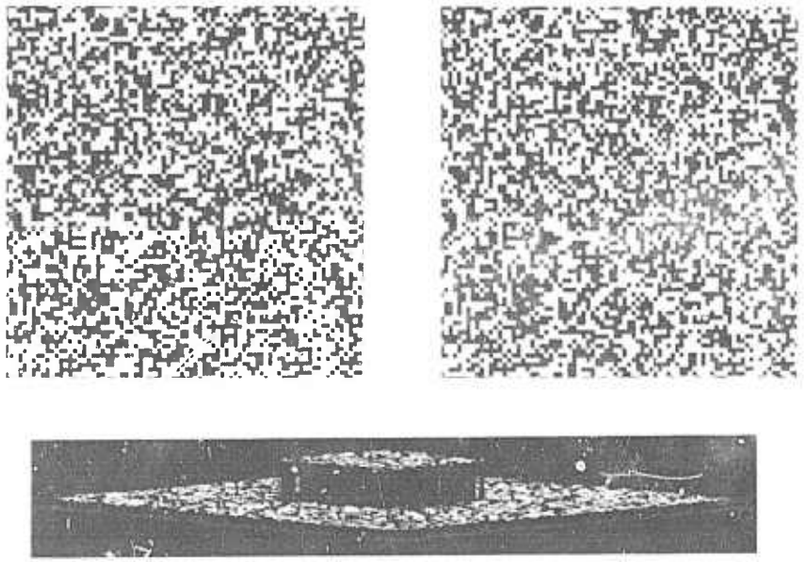


Figure 11. The top stereo pair is a 50% density pattern. The left image has and high pass filtered noise added to it so that the maximum magnitude of the noise is half the maximum magnitude of the image. The top disparity map is that obtained from the $w = 9$ channel, while the next disparity map is that obtained from the $w = 4$ channel. It can be seen that the $w = 4$ channel obtains a matching only in a few sections of the image. The bottom stereo pair is a 50% density pattern in which the left image has been compressed in the horizontal direction. The disparity map from the $w = 4$ is displayed below. It can be seen that the two planes are still evident, although the entire pattern appears slanted. This is in agreement with human perception.



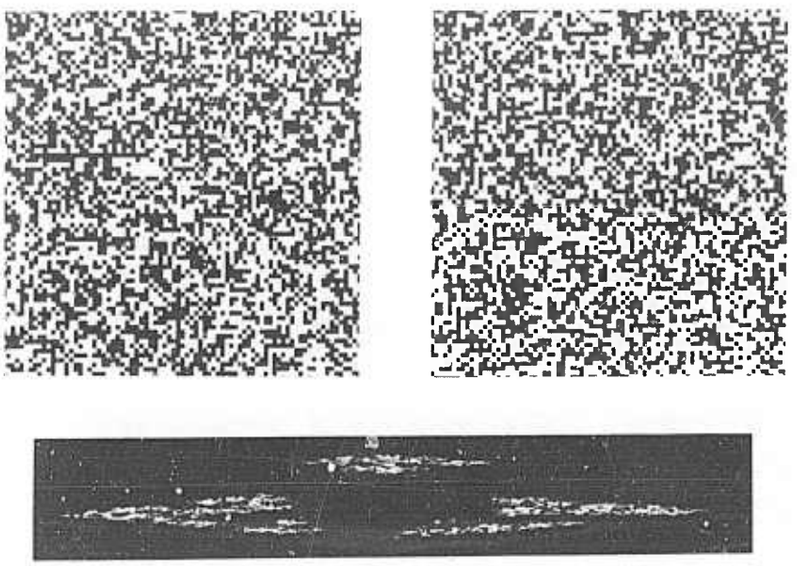


Figure 12. The top stereo pair is a 50% density pattern in which the left image has had 10% of the dots decorrelated. The disparity map is shown below. The bottom stereo pair is a 50% density pattern in which the left image has had 20% of the dots decorrelated. The disparity map is shown below. Note that in this case there are large regions of the image for which no match was made.

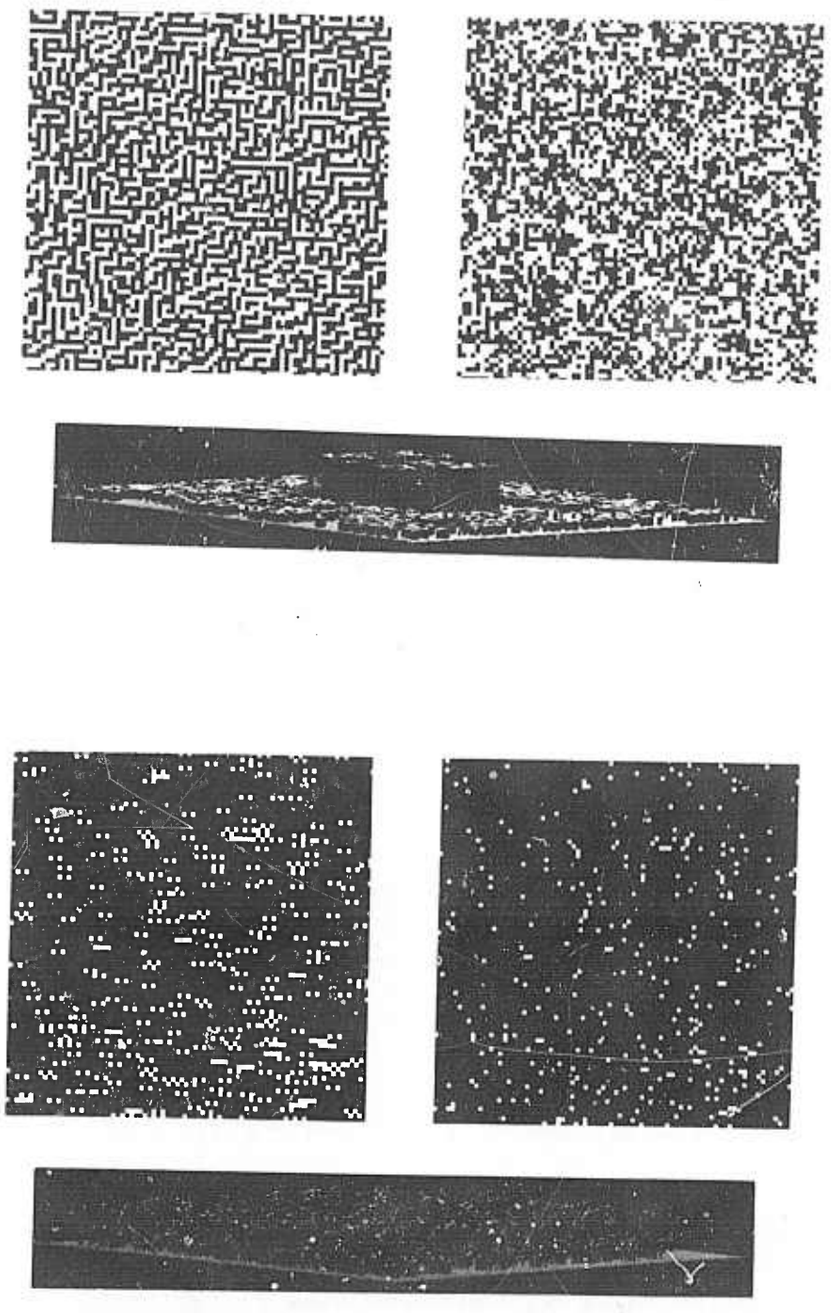


Figure 13. The top stereo pair is a 50% density pattern in which the left image has been diagonally decorrelated. Along one set of diagonals, every triplet of white dots has been broken by the insertion of a black dot, and along the other set of diagonals, every triplet of black dots has been broken by the insertion of a white dot. The disparity map is shown below. The bottom stereo pair is a special case of Panum's limit. The left image is formed by superimposing two slightly displaced copies of the right image. The disparity map is shown below, and consists of two superimposed planes.

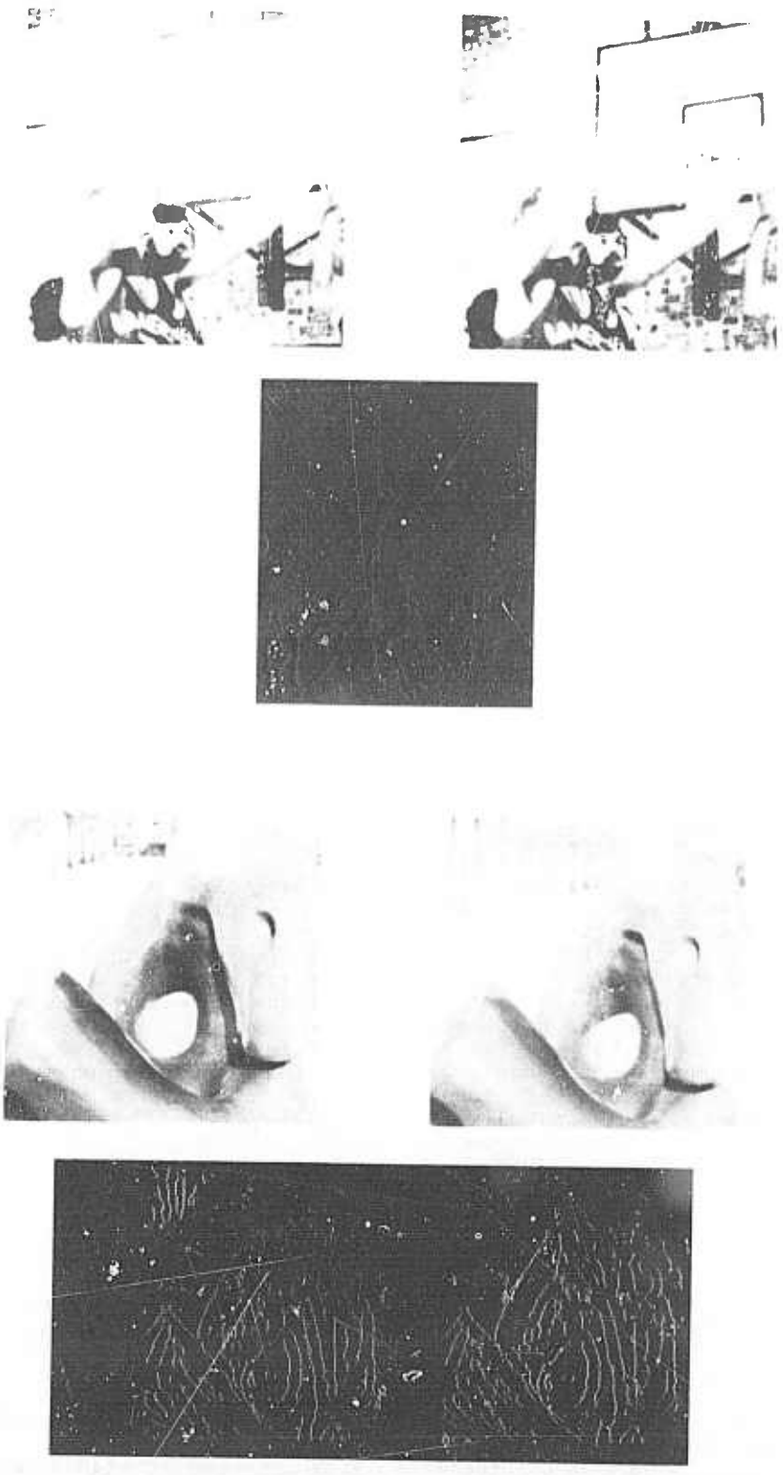


Figure 14. Examples of natural images. The top stereo pair is a scene of a basketball game. The disparity map below is viewed from the side, so that the width of the black bars indicates the relative disparity. The bottom stereo pair is of a sculpture by Henry Moore. The disparity maps below it are also viewed from the side. The left map illustrates the extreme range of disparity between the trees in the background and the sculpture itself. The right map has been adjusted to enhance the disparities of the sculpture, indicating its form.

EXPERIENCE WITH THE GENERALIZED HOUGH TRANSFORM

K. R. Sloan, Jr.
D. H. Ballard

The University of Rochester
Rochester, New York 14627

Abstract

The Hough Transform is a method for detecting curves by exploiting the duality between points on a curve and parameters of that curve. The initial work showed how to detect both analytic curves [Hough, 1962; Duda and Hart, 1972] and non-analytic curves [Merlin and Farber, 1975], in the case of binary edge images. This work was generalized to the detection of some analytic curves in grey level images, specifically lines [O'Gorman and Clowes, 1973], circles [Kimme et al., 1975], and parabolas [Wechsler and Sklansky, 1975].

Recently, the Hough technique has been extended to the detection of arbitrary non-analytic shapes in grey level images [Ballard, 1979]. This shape detection scheme has been implemented and tested on a variety of artificial images and has found application in the analysis of real aerial images. Experience to date indicates that the technique is robust with respect to occlusions, but requires reliable edge-element orientation determination.

1. Introduction

Shape is an important attribute of two-dimensional figures. In simple figure-ground binary images, the shape of the boundary of the figure is often the only interesting feature. We take "shape" to be a property of the entire figure, i.e., it is a global property.

Evidence about the shape of a figure is found at the boundary between figure and ground. Such evidence can be generated by the application of local edge-element detectors. An edge-element detector typically reports on the presence of an edge-element in a small window of an image, and on the orientation of that edge-element. Finding shapes in the image involves combining many pieces of local evidence into a global judgment.

The Hough Transform is a method for detecting curves by exploiting the duality betwqeen points on a curve and parameters of that curve. The initial work showed how to detect both analytic curves [Hough, 1962; Duda and Hart, 1972] and non-analytic curves [Merlin and Farber, 1975], in the case of binary edge images. This work was generalized to the detection of some analytic curves in grey level images, specifically lines [O'Gorman and Clowes, 1973], circles [Kimme et al., 1975], and parabolas [Wechsler and Sklansky, 1975].

Recently, the Hough technique has been extended to the detection of arbitrary non-analytic shapes in grey level images [Ballard, 1979]. Given an arbitrary shape, S, this generalized Hough technique provides a mapping from the orientation of an edge-element to the set of instances of S (as modified by location, rotation, and uniform scaling) which could have given rise to that edge-element. This mapping allows all local evidence for a particular instance of S to contribute to global decisions about the figure.

This shape detection scheme has been implemented and tested on a variety of artificial images and has found application in the analysis of real aerial images. Experience to date indicates that the technique is robust. Also, with appropriate "focus of attention" mechanisms, which are present in our implementation, the method is also efficient. However, the reliable determination of edge-element orientation is crucial to the success of this method.

2. Hough Techniques

All Hough techniques for shape detection consist of the following basic elements:

a) a local edge-element detector, E,
b) an n-dimensional parameter space, P, quantized and represented by an n-dimensional Accumulator Array, AA,
c) a mapping, M, from the information provided by E into P (and thus AA),
d) a voting rule, V, specifying how a particular edge-element affects the values of AA,
e) a Detection rule, D, specifying the conditions under which a particular shape has been detected.

Given these basic elements, shapes are found by the following procedure:

a) zero AA,
b) apply E everywhere in the image,
c) for each edge-element found, apply M to locate cells in AA. Then apply V to modify the contents of these cells. (i.e., vote for all possible "causes" of this edge-element),
d) finally, apply D to AA (choose the most popular shape).

Clearly, application of this technique depends on the ability to parameterize the shapes of interest, and the derivation of the mapping M from edge-element information to possible shape parameters.

Lines

The original Hough transform capitalized on the observation that straight lines can be completely specified by two parameters (e.g., an orientation θ, and a distance from the origin, ρ). What is more, the mapping, from a particular edge-element position to the set of straight lines it might be a part of, is easy to compute [Hough, 1962; Duda and Hart, 1972]. The idea is that an actual line in the image will give rise to many local edge-elements, all of which will "vote" for that line. Individual edge-elements will also vote for other lines, but the "correct" line will receive the most votes.

If the edge-element operator, E, provides directional information, then each edge-element maps to a unique line. Edge elements which line up vote for "their" line, and the line with the most visible edge-elements gets the most votes. Note that it is not necessary for the edge-elements to be connected (or even be near each other) in order that their votes reinforce one another - they must simply be colinear.

Circles

The description of circular figures in an image requires three parameters: x, y, ρ. The location of the center of the circle is given by $\langle x,y \rangle$ and the radius is given by the scale parameter, ρ. Once again, each edge-element in the image is evidence for a set of $\langle x,y,\rho \rangle$ triples.

If the direction of the edge-element is unknown, then the locus of points in parameter space representing circles which could have created this edge-element forms a right circular cone. In the presence of direction information, this locus is reduced to a line [Ballard, 1979]. As with line detection, circles which actually appear in the image will receive many votes; those which do not will receive few votes.

Arbitrary Shapes

The Hough technique can be extended to analytic shapes for which the mapping from edge-element to a locus of points in parameter space can be derived. Given certain assumptions about the meaning of "shape", we can also extend the technique to arbitrary, non-analytic shapes.

Consider a particular figure (e.g., an ellipse centered at $\langle 1,2 \rangle$ with its major axis parallel to the x-axis and of length 10, and its minor axis of length 5). Now, consider the set of figures which can be produced by translating, rotating, and uniformly scaling the original figure. For our purposes, all of these figures have the same shape.

The parameter space which captures this notion of shape is:

$$P = \langle x, y, \rho, \theta \rangle$$

where

$\langle x,y \rangle$ is the origin of a local co-ordinate system

ρ is a scale factor

θ is a rotation about $\langle x,y \rangle$.

This is the parameter space used in our generalized Hough Transform. Note that the Hough-spaces developed above for lines and circles are sub-spaces of P.

The key to all Hough techniques is the mapping from edge-element information to a locus of points in P. We assume an edge-element operator which provides directional information. As seen above, this directional information can drastically reduce the image of the

edge-element in P. Our mapping, M, depends strongly on the reliability of the edge-element direction.

Consider the hyperplane of P with $\theta = 0$, $\rho = 1$. We represent the mapping from edge-element location and orientation to figure location directly in an "R-Table" (see Figures 1 & 2). The orientation of an edge-element is used as an index into this table, where are stored a set of $\langle x,y \rangle$ vectors. When added to the $\langle x,y \rangle$ location of the edge-element in the image, these vectors point to possible locations for the origin of a figure's local co-ordinate system (its reference point). This map is easy to build, given an original master shape.

The expansion of the R-Table mapping to cover the remainder of P is performed dynamically by our voting procedure, V. This involves rotating the edge-element orientation before using it as an index into the R-Table, and scaling the R-Table entries thus formed before calculating the figure's hypothesized reference point.

3. Implementation and Experimental results

The generalized Hough Transform described above has been implemented and tested on a variety of artificial images and has found application in the analysis of real aerial images. Experience to date indicates that the technique is robust, given that the edge-element operator used to generate local evidence for the shape can provide reliable information about edge-element direction.

R-Tables

The R-Table defines the mapping from edge-element information (position and orientation) into a hyperplane of parameter space. This mapping is derived from an explicit master shape, in the form of a sequence of boundary points. Typically, we sketch (or trace) a shape. In order to ease the pain of carefully drawing a particular shape, we customarily sample the master shape boundary rather coarsely and then fill in the intermediate points using a B-spline fit to these points [Riesenfeld, 1973]. An arbitrary reference point is chosen for the origin of the local co-ordinate system.

Now, for each point on the master shape boundary, we calculate the orientation of the boundary edge-element at that point and the vector from the boundary edge-element to the origin of the local co-ordinate system. This is exactly an R-Table entry. The current implementation of the R-Table consists of a list of entries, tagged with the edge-element orientation, containing a lis of reference point vectors. Any scheme which associates edge-element orientation with reference point vectors will do.

Edge Detection

The examples shown below used a simple $3x3$ Sobel edge-element finder. In general, this is satisfactory. When, as in one example below, this does not provide reliable edge-element orientation, performance deteriorates seriously.

Detection criteria

For the purposes of these examples, the shape found by the generalized Hough Transform is determined by simply selecting the maximum value found in a smoothed (over a $3x3x3x3$ window) Accumulator Array. This does the right thing when, as in most of our examples, the maxima in the Accumulator Array are sharp peaks. For more problematic, noisy situations, clustering in parameter space may be required.

Artificial Images

Figure 3a-3f illustrates a few of the features of the experimental implementation of the generalized Hough Transform. These artificial images provide controlled conditions for our testing. In Figure 3a and 3b we see that, as expected, the method has no difficulty in finding the central shape at arbitrary scale and orientation (The black dots show the shape, as drawn from the R-Table and the parameter choices which received the most votes in the Accumulator Array, the central black dot is the reference point.) In Figure 3c we see what appears to be the same shape, obscured by another. Figure 3d demonstrates that there is enough evidence for the desired shape to correctly determine its location, orientation, and scale.

All of these images, of course, have very clean edges and the $3x3$ Sobel operator has no difficulty in correctly determining edge-element orientation. By way of contrast, see Figure 3e. In this image, which has been degraded by the addition of considerable noise, the edge-elements found by the $3x3$ Sobel operator are too short, and the noise hopelessly jumbles the orientation information. As a result, the generalized Hough Transform (which depends strongly on the accuracy of edge-element orientation) is unable to locate the shape. The guess shown is not much more than that - the Accumulator Array has no very strong peak, and we simply show the shape instance which received the most votes in a very

close election.

Aerial Photographs

The location of arbitrary, non-analytic shapes is not merely of interest in artificial images such as that shown above. The original version of the shape found above came from the aerial image shown as Figure 3f. Even the experimental version of the generalized Hough Transform has no difficulty in locating the pond in this image.

4. Focus of attention

One of the difficulties encountered in the application of this technique to real images, for the location of real shapes, is that the area searched for evidence of boundaries (the application of the edge-element detector and the mapping from edge-element information to parameter space) and the size of the parameter space can quickly become very large. The solution to these problems is, of course, to attempt to focus attention where possible.

Where to Look

One way to focus attention during the application of this shape-finding technique is to constrain the area searched for evidence of the figure's boundary [Russell and Brown, 1978; Russell, 1979]. In a system which routinely applies an edge operator over the entire image, this may not seem to be a solution (or even a problem). However, even after the edge-elements have been found, it is still necessary to apply the mapping to parameter space (one per desired shape). Our implementation includes the usual "bounding rectangle" limitation on the area in which edge-elements are to be found and mapped to parameter space. This improves performance significantly.

What to Look For

The second obvious way to focus attention is to constrain the objects being sought. Of course, a single application of the generalized Hough Transform concentrates on the location of a particular class of shape (that defined by the R-Table). In addition, it is usually possible to constrain the permissible values for some (if not all) of the parameters. Constraining the location of the reference point is related to the question of "Where to look". Constraining the parameters of scale or rotation is also possible, and certainly worth doing. Sometimes, the unconstrained search for a particular shape (such as the pond in Figure 3f) will result in almost complete information about the range of values to be considered in successive searches. For example, once the pond has been located (in parameter space, including location in the image, rotation and scale) map-like knowledge about this particular part of the world would allow the search for other shapes in the scene to be almost completely determined. Thus, our technique can both generate and benefit from such constraints.

Although our current implementation uses only a simple "bounding rectangle" constraint on the area of the image to be searched for boundary information, it is possible to combine information about the range of locations for the reference point, scale, and rotation. When all of these are sufficiently constrained, then the R-Table itself provides pointers to the locations to be searched in the image for edge-elements.

5. Conclusion

Shape is an important defining feature of many image objects, often the only useful feature. The key ideas behind the Hough Transform have been extended to produce a shape detection technique which performs well in the presence of occlusion, even for completely arbitrary, non-analytic shapes. As has been demonstrated, however, the technique depends strongly on the reliable estimation of edge-element orientation.

REFERENCES

Ballard, D.H. and Sklansky, J., A ladder-structured decision tree for recognizing tumors in chest radiographs, IEEE Transactions on Computers, Vol C-25, 1976, pp. 503-513.

Ballard, D.H., Generalizing the Hough Transform to Detect Arbitrary Shapes, TR55, Computer Science Department, University of Rochester, October, 1979 (a); submitted to Pattern Recognition.

Duda, R.O. and Hart, P.E., Use of the Hough transform to detect lines and curves in pictures, CACM 15, 1, Jan. 1972, pp. 11-15.

Hough P.V.C., Method and means for recognizing complex patterns, U.S. Patent 3,069,654, 1962.

Kimme, C., Ballard, D.H., and Sklansky, J., Finding circles by an array of accumulators, CACM 18, 1, Feb. 1975, pp. 120-122.

Merlin, P.M. and Farber, D.J., A parallel mechanism for detecting curves in pictures, IEEE Transactions on Computers, Vol C-24, Jan. 1975, pp. 96-98.

Gorman, F., and Clowes, M.B., Finding picture edges through collinearity of feature points, Proc. Third IJCAI, 1973, pp. 543-555

Riesenfeld, R., Application of B-Spline approximation to geometric problems of Computer-Aided-Design., UTEC-CSs-73-126, University of Utah, March, 1973.

Russell, D.M., Where do I look now?: Modeling and inferring object locations by constraints, Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, August, 1979.

Russell, D.M. and C.M. Brown, Representing and using locational constraints in aerial imagery, Image Understanding Workshop, November, 1978.

Sklansky, J., On the Hough technique for curve detection, IEEE Transactions on Computers, July, 1977.

Shapiro, S.D., Transformation for the computer detection of curves in noisy pictures, Computer Graphics and Image Processing, 4, pp.328-338, 1975.

Shapiro, S.D., Properties of transforms for the detection of curves in noisy pictures, Computer Graphics and Image Processing, 8, pp. 219-236, 1978.

Wechsler, H. and Sklansky, J., Automatic detection of ribs in chest radiographs, Pattern Recognition, 9, Jan. 1977, pp. 21-30.

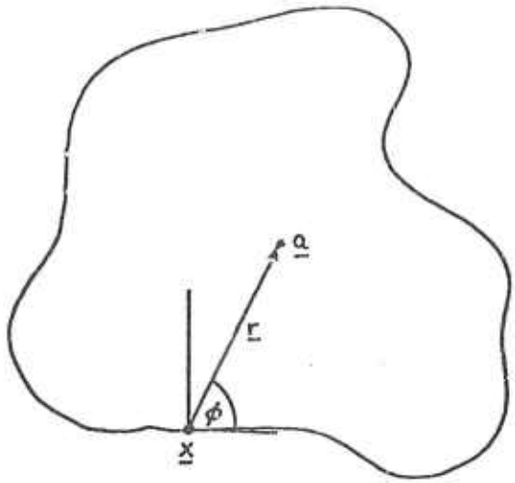


Figure 1: Geometry for Generalized Hough Transform.

i	ϕ_i	R_{ϕ_i}
0	0	$\{\underline{r} \mid \underline{a} - \underline{r} = \underline{x},\ \underline{x} \text{ in } B,\ \phi(\underline{x}) = 0\}$
1	$\Delta\phi$	$\{\underline{r} \mid \underline{a} - \underline{r} = \underline{x},\ \underline{x} \text{ in } B,\ \phi(\underline{x}) = \Delta\phi\}$
2	$2\Delta\phi$	$\{\underline{r} \mid \underline{a} - \underline{r} = \underline{x},\ \underline{x} \text{ in } B,\ \phi(\underline{x}) = 2\Delta\phi$
...	...	...

Figure 2: R-table format.

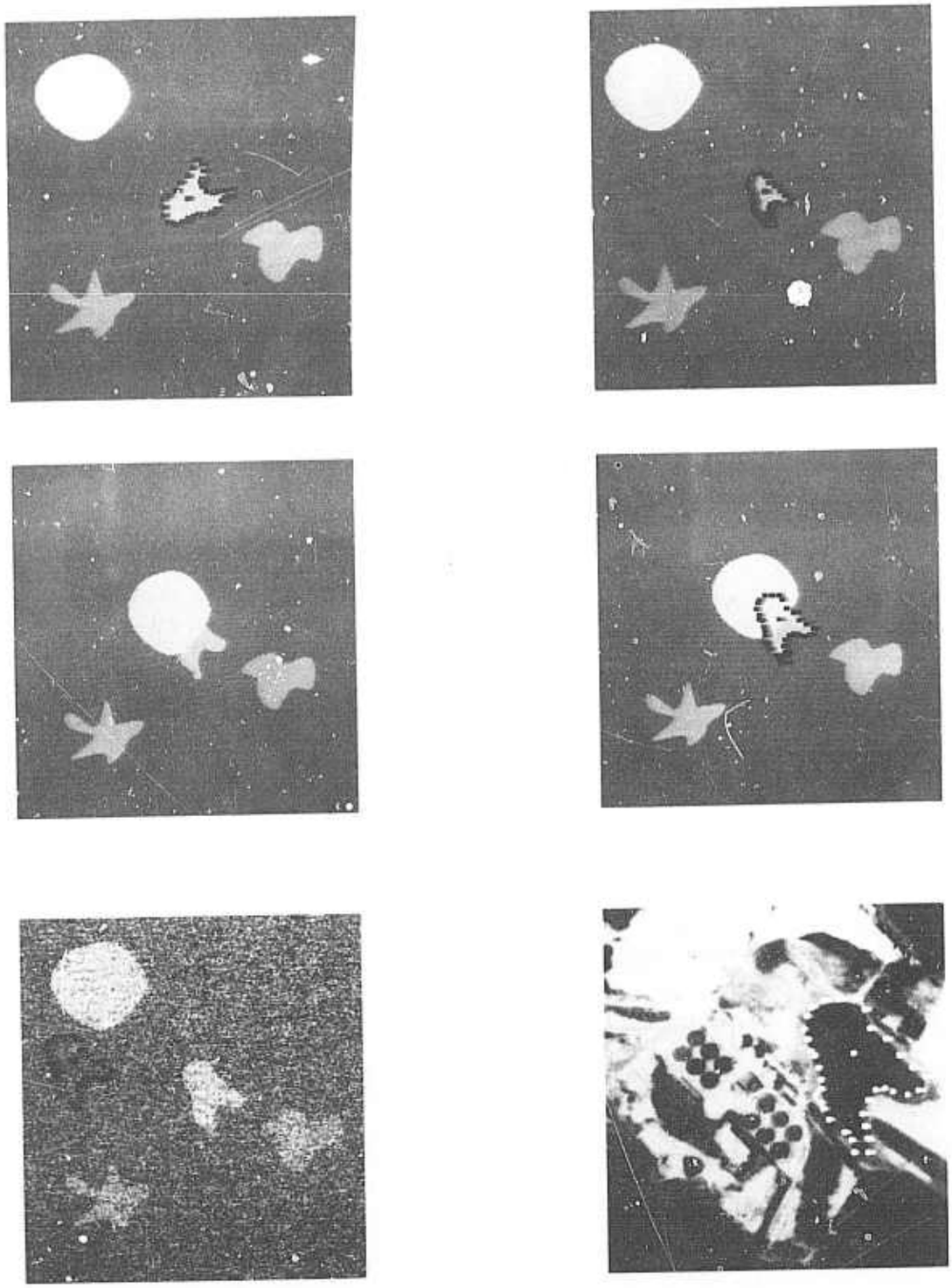


Figure 3

THE GAUSSIAN SPHERE: A UNIFYING REPRESENTATION OF SURFACE ORIENTATION

John R. Kender

Department of Computer Science
Carnegie-Mellon University, Pittsburgh, PA 15213

ABSTRACT

This is an informal, qualitative presentation of some recent results concerning the representation of surface orientation. We show that the existing methods using the gradient space are inadequate in several ways. Using the Gaussian sphere instead greatly simplifies the mathematics of many surface orientation problems. One intuitively satisfying and pedagogically transparent result is that many existing relations map into latitude lines or longitude lines on an appropriately oriented Gaussian sphere. This use also leads to several predictions concerning what surface orientation properties are the easiest to calculate; they also seem to be the most important. Some problems still remain, however. They may be better solvable through the application of spherical geometry.

INTRODUCTION

The representation and manipulation of surface orientation is a critical part of image understanding. Any domain that has a non-negligible depth component requires some method of dealing with changes in depth, both sudden and smooth. Smooth depth changes give rise to surfaces; surface orientations (local depth changes) have two degrees of freedom. That is, surfaces can tilt top-to-bottom, and left-to-right.

On one hand, representing these degrees of freedom is not a problem. As long as smooth changes in orientation are smooth in the representation, and as long as the space is complete, one space is mathematically equivalent to any other. On the other hand, spaces are not all mathematically simple, nor are they representationally concise. We will show that the gradient space, currently the preferred representation, is not as elegant nor manipulable as the Gaussian sphere parameterization. We will also show, by using the spherical representation, that many existing results concerning surface orientation have the same particularly simple form.

In this paper, many of the mathematical details are suppressed for the sake of presentation. However, they appear in full detail in the author's thesis [Kender, 1980]. The stress here is on qualitative aspects of the two representation spaces; this is also reflected in the qualitativeness of the figures.

INADEQUACIES OF THE GRADIENT SPACE

The gradient space represents surface orientations in terms of the rate of change of surface depth with respect to two orthogonal image axes. Basically, the representation uses the gradient resolved into its two component vectors.

Given the imaging geometry that places the observer on the negative z axis and the film on the plane z = 1 (see Fig. 1), let any surface be represented by the equation $z = px+qy+c$. Then the rate of change of depth along the x axis is p, and along the y axis it is q. By the usual definition, the gradient is (p, q). (For more detail, see [Huffman, 1971] or [Horn, 1977])

The gradient space has several elegant properties. As the image rotates about the line of sight, the gradient space representation of it similarly rotates. The gradient space is a dual space to the image space, in which planes are represented by points, and vice-versa; image lines are mapped into (different) gradient lines. This simplifies many representational problems. Scene properties such as convexity and concavity are also neatly reflected in simple gradient space relations.

Intrinsic Inadequacies

However, the gradient space is only half a space. It cannot represent surfaces oriented away from the viewer's main line of sight. Although this is annoying if the image is taken under orthographic projection (the usual assumption), it is absolutely critical under perspective. Under perspective, it is possible to actually see "behind" a surface (see Fig. 2). Such surfaces are not representable in the gradient space at all; the fundamental cause is the gradient's indifference to surface "sidedness". Under orthography questions of sidedness never occur. Under perspective they are rampant and indispensable.

Even strictly under orthography, other related intrinsic problems occur. For example, if the gradient space is used for work with reflectance, it cannot represent the case where the illuminant is in front of the camera (shooting into the sun, say).

It is not easy to patch this deficiency in the gradient space. It is possible to conceive of a "negative" gradient space, like the existing "positive" one, and sutured to it at infinity. As a surface became infinitely steep, its gradient representation would "cross over" into the negative space as soon as the surface turned away from the observer. Light sources would do the same as they moved to in front of the camera. However, there are solutions that are far more pleasing.

Inadequacies in Application

The gradient space is also difficult to work with in practice. Since by definition it is a coordinate system based on derivatives, it is an infinite space. There is no straightforward way to deal with its vast expanse in a computer program.

It is a misshapen space as well. Within its unit circle are represented all those planes whose overall orientation away from the viewer (gradient magnitude) are less than 1. Planes sloping as much as 45 degrees are represented there. However, very little happens outside this tight little circle; most of the space is effectively wasted. Almost all of it is devoted to distinguishing between planes that are small fractions of a degree away from being parallel to the line of sight. There is no straightforward way to code around this, either.

ADVANTAGES OF THE GAUSSIAN SPHERE

Consider now the Gaussian sphere representation of surface slope. Intuitively, it can be imagined as a transparent globe about the viewer's head, somewhat like an oversized fishbowl. It is clear that the surface of the globe is a two-dimensional space (a manifold); it can be mapped one-to-one onto the gradient space. However, we will treat it differently. Given a plane, represent its orientation by that unique point on the sphere the plane would touch if it were rigidly translated there through space. Such a representation has sidedness; opposite sides of a surface touch opposite sides of the sphere.

More exactly, the Gaussian sphere identifies orientation not with surface gradient, but with a type of surface normal. (The normal is "aware" of which side of the surface the object is on.) Mathematically, it normalizes the normal vector to length 1, instead of normalizing the gradient to a vector of the form (p, q, 1). Three-space vectors of length 1 are mapped on a sphere of unit radius (the "Gaussian" sphere) in the straight-forward way: the point on the sphere has the same coordinates as the normal does.

Now center the sphere at the image origin. Consider the z axis as the symmetry axis of a spherical coordinate system, and the x axis as the usual theta axis. All surface orientations are now in spherical coordinate form. Since the radius is uniquely 1, drop it entirely, just as the third coordinate of the gradient is usually dropped. Instead of (p, q), we can use (theta, phi). The Gaussian sphere thus induces a two-angle representation system that incorporates all possible orientations and directions with respect to the observer.

Preservation of Good Gradient Space Properties

So far, nothing has been lost, and some has been gained. All the good properties of the gradient space can be shown to be preserved. The Gaussian space is smooth. If the image rotates, it also rotates as the gradient space does. The dual character of the space remains, and even the concave-convex relations hold, suitably modified.

Furthermore, if the image tilts (in the strict sense of the word: if it rotates about the x axis), the representation tilts isometrically about the sphere surface, too: a new property. It is also now very easy to find all the planes that are perpendicular to a given plane. Graphically (Fig. 3), find the "equator" to the given plane's "pole". The great circle is the locus of the representations of the planes perpendicular to the plane represented by the isolated point.

Graphic Equivalence

Although the Gaussian sphere is more powerful than the gradient space, the two are closely related. The gradient space is easily shown to be the projection of the Gaussian sphere from its center to one tangent plane. Under these conditions, lines in the gradient space are mapped from great circles on the sphere.

As Fig. 4 shows, it is clear why the gradient space is only half a space: a second plane, parallel to the first, would be needed to represent all orientations. (This second plane is the "negative" gradient space). This same basic projection is the key to the following observations.

THE SPHERE AND SHAPE FROM SHADING

Reflectance maps represent the way observed brightness is dependent on surface orientation, given the position of the illuminant. In at least two well-studied cases, reflectance maps have simple representations on the Gaussian sphere. Further, the spherical representation extends existing results.

Under the assumptions of lambertian reflectance, the reflectance map forms a series of nested conic sections. Horn has observed that the family of curves corresponds to the cross-section of a properly nested set of cones [Horn, 1977]. However, it is not very difficult to prove a firmer result. It can be shown that the cones arise by being the projection of a family of latitude lines on the Gaussian sphere. The "North Pole" is the illuminant direction, towards which the pole is tilted. (See Fig. 5.) The percentage of reflected light is simply the sine of the latitude measured in degrees, with respect to the pole. Thus, equal spacing of latitude lines yields equal differences in reflectance. The equator (a line in the gradient space) corresponds to those surfaces just beginning to be self-shadowed.

Consider the assumption that reflectance is related to incidence and emergence angles by the value cos(i)/cos(e) This is the so-called lunar reflectance relation [Horn, 1977]. Under these conditions, the reflectance map consists of parallel lines. On the sphere, these map into longitude lines. The self-shadowing line is again the line of perpendiculars to the light direction; this time, however, the longitude lines are parallel to it. An extra advantage of this representation is that it is easy to find the line along which reflectance is unity (maximal reflectance). It is simply half the angle (but not half the gradient space distance) from the origin to the light direction.

The gradient space method cannot represent illuminants in front of the observer. Not only can the

spherical system represent them, but it can be shown that the representation is smooth and straight-forward. Essentially, the reflectance always arises from latitude lines, even if the sphere is pointed away from the observer to the light source; there is no difference in the mathematics.

THE SPHERE AND SHAPE FROM TEXTURE

Normalized texture property maps convey information very much like reflectance maps; in fact, the latter can be considered are special cases [Kender, 1979]. Under many circumstances the texture maps are also made very simple under the Gaussian representation.

If the textural element chosen for the determination of surface orientation is an image slope element, and the texel-texel relationship is parallelism, then the results are especially simple. This is the situation that occurs with tiled floors, brick walls, hairbrushes, etc. Under perspective, the curves that are generated in the gradient space are again a conic family, as with lambertian reflectance. However, the parameter that distinguishes one curve from another is the texel-surface relation.

Suppose one assumes that a surface is perpendicular to the textural objects defining it. This is the case of, say, finding the ground plane from the orientation of vertical trees. Then two texels generate, through their normalized maps, a polar point: this is, in fact, the vanishing point of the texture itself. This point orients the sphere, as if the point were a light source. If, however, the texels are assumed to lie in the plane they define, the curve generated is a line in the gradient space (an equator on the sphere) that is analogous to the line of self-shadow. Intermediate texel-surface relations generated intermediate curves; the sine of the angle between texel and surface is analogous to reflectance. The relation is represented by the "hairy sphere" of Fig. 6. Note that the image of the texels form longitude lines that point to the image of the pole in the gradient space.

The analogue between parallelism under perspective and lambertian reflectance is not quite exact. The reflectance map stops at the line of self-shadowing, and a given reflectance value generates only one curve. However, a given intermediate-value texel-surface relation generates two full conic sections: one each for the two ends of a texel that can contact the sphere. For a full analogy, we would need for the reflectance map an anti-sun emitting darkness from a point exactly opposite the true illuminant position.

Other relationships also map into simple Gaussian representations. Area under orthography is identical to the map of lambertian reflectance with the illuminant at the observer. That is, both are again latitude lines, with the sphere pointed straight at the origin. (The gradient space has concentric circles about the origin). Length under orthography maps into longitude lines similar to those of the lunar reflectance case. The actual normalized properties differ in size, though the orientations of the longitudes are purely analogous.

Other relations, a bit too complex to describe here, also exist; again, many map into either latitude or longitude lines on an appropriately tilted sphere.

THE SPHERE AND OTHER SHAPE METHODS

Woodham has shown that assumptions about occluding contour can be exploited in the determination of surface shape [Woodham, 1978]. Specifically, assume an object to be a right generalized cone: that is, its cross sections are circles. Then information from the silhouette is sufficient to create constraints in the gradient space that can be used to derive local surface orientation.

Unfortunately, his results are expressed in an object-centered, parametric form. Both p and q are given in terms of very complex functions of an angle internal to the object; this angle is measured about the generalized cone's axis. It can be shown, however, that when converted to the Gaussian representation--a non-trivial task--the restraint curves are again latitude lines on the sphere! Further, the sphere axis is aligned parallel to the generalized cone axis. Thus, tilting the cone is simply paralleled by tilting the sphere; the corresponding transformation in the gradient space is very involved.

One last example of using the sphere deals with motion. Prazdny has noted that if one had a retina that was hemispherical, with its focal point at its center, then the visual flow lines would have a simple form [Prazdny, 1979]. They, too, would be longitude lines (for translations), and latitude lines (for rotations). Although the retina is not hemispherical, the imagined fishbowl about the observers head has the required properties: the focal point at the center is served by the observer himself.

GENERAL COMMENTS

In all the examples given, the Gaussian sphere simplified a relation involving surface orientations; in all cases they were reduced to families of either latitude lines or longitude lines. Each example also exhibited a simple property for orienting the Gaussian sphere; the rest fell out naturally once the pole orientation was found. Likewise, knowing the image of the equator also is sufficient; the pole is easily derived from it. This would suggest that determining the orientation of the pole (i.e. the light direction, the generalized cone axis, the vanishing point, etc.), would be most useful to an image understanding system.

There is a bit of evidence to believe this. With regard to shading, the polar point is the direction of maximum reflectivity, and the equator is the line of self-shadow. One finds that in line drawings, highlights and shadow lines are regularly drawn in, even if the drawing is not shaded.

Further, most smoothly curved, illuminated objects have a "terminator", a line where the shadow begins. It is undetectable with edge detectors, since it is not a step-discontinuity in brightness. The human eye, however, is sensitive to changes in the second derivative of brightness; these changes characterize the terminator. Perhaps this is one reason for the ability.

With regard to texture, finding the pole would be especially easy under the assumptions of texels being either perpendicular or parallel to the surface they define--but not in between. This seems to be the case: it appears to be

very difficult to perceive textures formed from oblique texels.

FUTURE RESEARCH

The Gaussian sphere does not simplify everything. In particular, it does poorly with relationships involving perpendicularities. This appears to be due to the one-to-many mapping of a plane to its perpendicular counterparts. (This problem is shared by the gradient space, too.) It also is inelegant on textures involving perpendicularity, as in Kanade's skewed symmetry method [Kanade, 1979]. Although the curves traced on the sphere by this method are well-known (they are "sphero-conics"), they are far from elegant.

Many areas are simply unexplored. Is it the case, say, that generalized reflectance functions are also simpler on the sphere?

It increasingly appears that the most promising way to approach problems of surface orientation is through spherical geometry. Most surface orientation methods require the intersection of loci in the representation space. Capturing the important properties of such curves under the very different conditions of spherical geometry may lead to further insights into the simplification of their representation and manipulation.

REFERENCES

B. K. P. Horn, "Understanding Image Intensity," Artificial Intelligence, Vol. 8, 1977.

D. A. Huffman, "Impossible Objects as Nonsense Sentences," Machine Intelligence 6, R. Meltzer and D. Michie (eds.), Edinburgh University Press, 1971.

T. Kanade, "Recovery of the Three-Dimensional Shape of an Object from a Single View," Technical Report, Carnegie-Mellon University, 1979.

J. R. Kender, "Shape from Texture: A Computational Paradigm," Proceedings of the ARPA Image Understanding Workshop, Science Applications Inc., Apr., 1979.

J. R. Kender, "Shape from Texture," Ph.D. Thesis, Computer Science Dept., Carnegie-Mellon University, 1980 (forthcoming).

K. Prazdny, "Egomotion and Relative Depth Map from Optical Flow," Ph.D. Thesis, Computer Science Dept., University of Essex, 1979.

R. Woodham, "Reflectance Map Techniques for Analyzing Surface Defects in Metal Castings," Ph.D. Thesis, Massachusetts Institute of Technology, 1978.

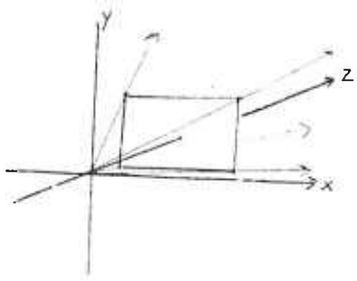


Fig. 1. The Imaging Geometry

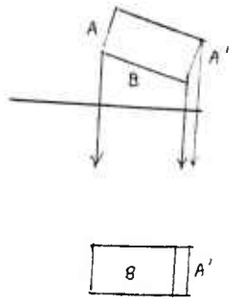


Fig. 2. Seeing "Behind" Using Perspective

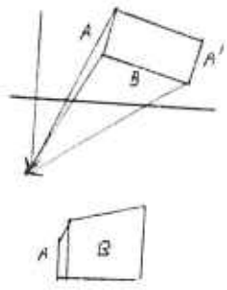


Fig. 3. Finding Perpendiculars

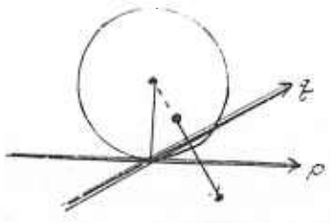


Fig. 4. Gaussian-Gradient Equivalence

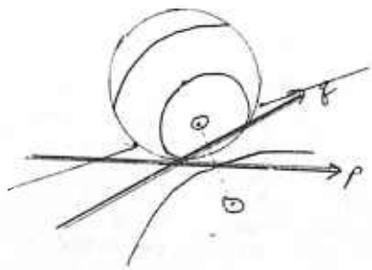


Fig. 5. Conic Family from Latitudes

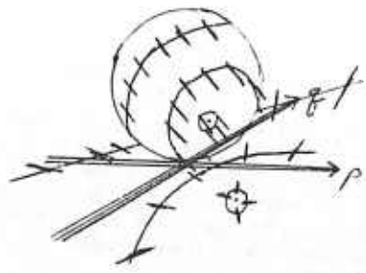


Fig. 6. Textured Sphere

OBJECT DETECTION AND MEASUREMENT USING STEREO VISION

Donald B. Gennery

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

A method of detecting and measuring objects for the purpose of representing three-dimensional outdoor scenes, using data such as that obtained from stereo vision or from a scanning laser rangefinder, is described. Objects are approximated by ellipsoids. Segmentation of the objects from the background and from each other is done by finding the ground surface, forming a preliminary segmentation by clustering the points above the ground by more than a threshold, fitting ellipsoids to match these clusters and to avoid obscuring the other points, and adjusting the clusters according to the fits. The method is designed to produce results useful for obstacle avoidance and navigation in an exploring vehicle, such as a Mars rover. An example is given showing results obtained from a stereo pair of pictures of Mars from the Viking Lander.

1. Introduction

In a previous paper [1], a stereo vision system and its possible use in an exploring vehicle was described. This paper describes further work towards a complete system for an exploring vehicle, specifically, the use of three-dimensional data for the detection of objects and the measurement of their position, size, and approximate shape. Although the three-dimensional data could be obtained from a laser rangefinder, the object detector is designed to be tolerant of errors in this data, such as mistakes produced by incorrect matches in stereo vision data and poor accuracy of distances from stereo.

Many approaches are possible in describing the shapes of objects. For example, generalized cones [2] can be used to describe complicated objects. However, in some cases the resolution of the data is insufficient to produce detailed information about the shape. In other cases, the objects are so irregular as to make such detailed descriptions very difficult. However, in such cases information about the position and size and some crude information about the shape may still be quite useful. For example, for obstacle avoidance in a roving vehicle, this sort of information is adequate. Furthermore, this sort of information for each object in a large scene containing many objects amounts to quite detailed information concerning the whole scene, and thus it would be useful for navigation.

For these reasons, and because of its convenient mathematical properties, here each object will be approximated by an ellipsoid. By "object" we do not necessarily mean an actual physical object, but merely a portion of the scene that can be reasonably approximated by an ellipsoid. Thus, if we use as an example a vehicle exploring Mars, an object may be a single rock on the Martian surface, two or more adjacent rocks, or merely a bump in the ground. Also, an L-shaped physical object might be represented as two objects.

This ellipsoidal representation should be quite appropriate for representing rocks on Mars, because rocks probably tend to resemble more nearly ellipsoids than any other simple shape. However, it could also be used to represent cars in a parking lot or trees in a field, for example, especially in aerial photographs where the resolution may be poor compared to the size of the objects, and in other cases where precise object description or recognition is not necessary but rather an overall description of the scene is desired.

The stereo vision processing or laser rangefinder results in data representing the three-dimensional position of a large number of points distributed over the scene. The first step in the processing of this three-dimensional data is to find the ground surface. A method of doing this was previously described [1]. In general, an entire scene would be partitioned into small areas, in each of which the ground would be approximated by a plane or paraboloid. Then points which are above the ground by a sufficient amount (depending on the computed accuracy of the points, the roughness of the ground, and the minimum size of object that is of interest) are candidates for points on objects.

These above-ground points are clustered to produce preliminary groupings of points which correspond roughly to objects. An ellipsoid is fit to each cluster by first computing an initial approximation based upon the moments of the points in the cluster and then iterating a weighted nonlinear least-squares adjustment to fit the ellipsoids to these points and to avoid obscuring other points. Then, according to the relative positions of the ellipsoids and points, clusters can be broken or merged, and the process repeats until the apparently best segmentation is found. Each of these steps will be described in the following sections.

The object detection and measurement process as described here uses only the three-dimensional position information. The brightness information is discarded after the stereo processing. However, a more complete system would use both types of information. Perhaps an edge detector could be applied to the brightness data in the regions near the outlines of the ellipsoids in order to refine the boundaries of the objects, for example.

Matrix notation will be used throughout this paper. Matrices will be denoted by capital letters. The transpose of a matrix A will be denoted by A^T, and the inverse of A will be denoted by A^{-1}. The trace of a square matrix A (sum of the diagonal elements, which is equal to the sum of the eigenvalues) will be

denoted by tr(A). A vector in three-dimensional space will be represented by a 3-by-1 matrix containing the Cartesian coordinates, usually denoted by X with an appropriate subscript. (Hohn [3] provides a good text on matrix algebra.)

2. Preliminary Clustering

Once the ground surface has been determined, all points that are above this surface by more than a threshold are clustered to form an initial approximation to the segmentation of the scene into objects.

Various clustering techniques could be used here. One possibility is a relaxation method, such as Zucker's [4]. However, at present, the clustering is done by using the minimal spanning tree of the points. (The minimal spanning tree is the tree connecting all of the points such that the sum of the edges is minimum.) This is computed by using the nearest neighbor algorithm [5]. (The length of the edges of the tree is defined here as the three-dimensional Euclidean distance between the points.) Then the tree is broken at every edge whose length is greater than twice the average length of the adjacent edges [5]. However, a minimum length for an edge to be broken (related to the resolution of the data) is specified, so that the method will not be overly sensitive to local fluctations in the data. Also, a maximum can be specified, beyond which all edges are broken.

3. Initial Approximations to Ellipsoids

Since each ellipsoid will be fit to a cluster of points by an iterative process, an initial approximation is needed. A good approximation increases the likelihood of convergence, decreases the number of iterations required, and can be used as the result in case the iterations do not converge. This initial approximation is obtained from the three-dimensional moments, through the second order, of the points in the cluster.

An ellipsoid can be represented by the following matrix equation:

$$(X-X_c)^T W(X-X_c) = 1$$

where X is a vector of the three-dimensional coordinates of any point on the surface of the ellipsoid, X_c similarly is the position of the center of the ellipsoid, and W is a positive-definite symmetrical 3-by-3 matrix. (See, for example, Hohn [3].) Let M denote the inverse of W. (The square roots of the eigenvalues of M are the lengths of the semi-axes of the ellipsoid.) The relationship between the computed moments and the matrices X_c and M depends on the distribution of points over the ellipsoid. If the object has been viewed from a single point, we will have points distributed nonuniformly over half of the surface. (Actually slightly less than half will be seen because of perspective. Also, in stereo vision, both cameras must see each point, so that with a single pair of cameras only the common area seen from both camera positions will appear. These two effects will be neglected below, however.)

We assume here that the object is seen from a single viewpoint by a raster scanning device which produces points distributed uniformly in the image plane. Such a device might be a scanning laser rangefinder or an area-based stereo system. Actually, because of missing points, the distribution will not be uniform. It would be possible to estimate the actual distribution by computing higher-order moments, but this might be overly sensitive to randomness in the distribution or an inadequate density of points, so it is not attempted here. As an approximation, we assume an orthogonal projection instead of a central projection. Let X_s denote the vector of normalized first moments (centroid) and M_s denote the matrix of second moments about X_s obtained with this distribution, and let X_o denote the position of the camera.

The relationship connecting X_s and M_s to X_c and M can be derived by first considering the case of a sphere of radius r. A little integration shows that in this case the eigenvalue of M_s corresponding to the eigenvector X_o-X_c is $r^2/18$, the other two eigenvalues are both $r^2/4$, and X_s is X_c plus 2r/3 times the unit vector in the X_o-X_c direction. All three eigenvalues of M should be r^2 in this case. An ellipsoid can be considered to be a distorted sphere (using stretching and skew distortions). Thus the ellipsoid can be considered to be stretched in the various directions by the amount given by the square roots of the ratios of the above eigenvalues, but in computing the displacement of the center, instead of r the distance from X_c towards X_o to the ellipsoid surface must be used. Thus the displacement of the center is the vector $2(X_o-X_c)/3$ divided by the scalar $\sqrt{(X_o-X_c)^T W(X_o-X_c)}$. Since X_c, X_s, and X_o are colinear, X_c can be replaced by X_s without changing the value of this ratio. Also, W ($=M^{-1}$) can be replaced by $M_s^{-1}/18$ in this expression, because of the stretching discussed above. Thus X_c can be computed from X_s by translating by this amount. To compute M, we can take 4 times M_s to account for the factor of 4 in two dimensions, but this leaves 14 out of the factor of 18 by which we need to stretch the moments in the direction toward the camera. This extra amount can be added by adding 14 times the moment produced by a fictitous point at the intersection of the X_s-to-X_o line and the surface of the ellipsoid corresponding to M_s. In order to keep the ellipsoid to a reasonble shape when there are not enough points to determine it well, M as obtained above is averaged with a scalar matrix whose diagonal elements are s (which represents a sphere of radius $\sqrt{s}$), with the average weighted so that the sphere represents four additional points in the moment computation. The value of s is determined so that it is the average of the two components of the second moments at right angles to X_o-X_s, but limited by the average of all three components (the three eigenvalues), including the effect of 14 times the effect of the fictitious point, as above, as an upper limit, and excluding this effect, as a lower limit. This avoids putting undue weight on the X_o-X_s dimension when the ellipsoid is long in this direction, since this dimension is less reliable because of the factor of 18 compression.

By combining the above information, the computation of the initial approximation can be expressed as follows:

$$X_s = \frac{1}{n}\sum X_p$$

$$M_s = \frac{1}{n}\sum (X_p - X_s)(X_p - X_s)^T$$

$$M_t = 4M_s + \frac{14(X_o-X_s)(X_o-X_s)^T}{(X_o-X_s)^T M_s^{-1}(X_o-X_s)}$$

$$X_c = X_s - \frac{2\sqrt{2}(X_o-X_s)}{\sqrt{(X_o-X_s)^T M_s^{-1}(X_o-X_s)}}$$

$$s = \min\left[\max\left(\frac{4}{3}\,\mathrm{tr}(M_s),\ 2\,\mathrm{tr}(M_s) - 2\,\frac{(X_0-X_s)^T M_s (X_0-X_s)}{(X_0-X_s)^T(X_0-X_s)}\right),\ \frac{1}{3}\,\mathrm{tr}(M_t)\right]$$

$$M = \frac{n}{n+4} M_t + \frac{4}{n+4} sI$$

$$W = M^{-1}$$

where X_p represents any point in the cluster, n is the number of points in the cluster, the summations are over these points, and I is the 3-by-3 identity matrix.

4. Iterative Solution for Ellipsoids

The adjustment of the ellipsoids is done by a modified least-squares approach. Each ellipsoid is adjusted so as to minimize the weighted sum of the squares of two kinds of discrepancies: the amounts by which the points (usually points in the cluster being fit) miss lying in surface of the ellipsoid, and the amounts by which the ellipsoid hides any points as seen from the camera position. (In the latter case, the discrepancies actually should be considered separately for each camera that sees the point in question. However, for narrow-angle stereo we use as a reasonable approximation the assumption that the "camera" is at the midpoint of the stereo baseline.) Including the second kind of discrepancy is useful in helping to determine the size and shape of the object when the points on the object itself do not contain sufficient information. Also included in the weighted sum of squares to be minimized are *a priori* terms which tend to force the ellipsoid by default to become a sphere near the ground when the points do not constrain it well.

The first kind of discrepancy above optimally should be defined as the length of the normal from the point in question to the surface of the ellipsoid. However, computing this requires solving a sixth-degree equation. Therefore, as an expedient the distance between the point and the surface along a straight line from the center of the ellipsoid to the point is used instead. In order to be consistent with this definition, the second kind of discrepancy is defined as follows. The midpoint of the two intersections of the surface of the ellipsoid with a line from the camera to the point is first found. Then the discrepancy of the first kind is computed for this midpoint. Both kinds of discrepancies are illustrated in Figures 1 and 2.

Now we must consider exactly for which points which kind of discrepancy is computed. There are five regions of space to consider, according to whether the point is to the side of the ellipsoid as seen from the camera (that is, the line through the camera postion and the point does not intersect the ellipsoid), is in front of the ellipsoid as seen from the camera, is inside the front portion of the ellipsoid (in front of the surface of midpoints as defined above), is inside the back portion of the ellipsoid, or is behind the ellipsoid. Also, there are two kinds of points to consider, according to whether or not the point is in the cluster which is assumed to correspond to this object. This produces ten combinations in all, which are illustrated in Figures 1 and 2. They divide into four categories.

First, if the point is not in the cluster and is either in front of the ellipsoid or is to the side, there is no discrepancy and this point is not included in the computations.

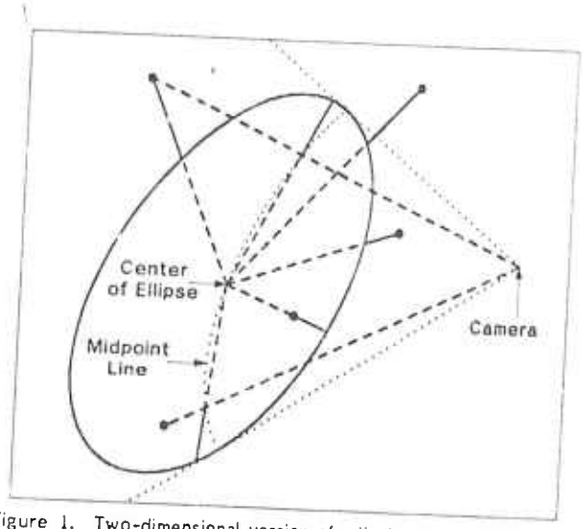


Figure 1. Two-dimensional version of adjustment, showing points belonging to this cluster. Solid dark straight lines are discrepancies to be minimized.

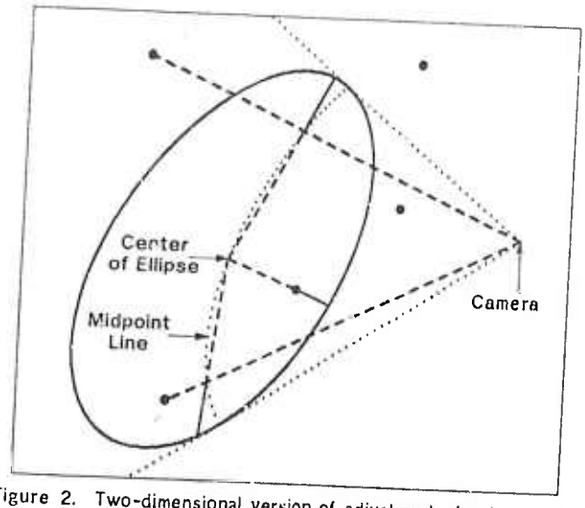


Figure 2. Two-dimensional version of adjustment, showing points not in this cluster. Solid dark straight lines are discrepancies to be minimized.

Second, if the point is in the cluster and is either in front, inside the front half, or to the side, or if the point is not in the cluster and is inside the front half, the first kind of discrepancy is used.

Third, if the point is not in the cluster and is behind the ellipsoid, or if either kind of point is inside the back half, the second kind of discrepancy is used.

Fourth, if the point is in the cluster and is behind the the ellipsoid, both kinds of discrepancies are used, and the point acts as two points in the computations. This is because there are two separate components of error in this case: the object does not extend enough on the side to hide the point, but it apparently bulges out in back (relative to the ellipsoid) to include the point.

In order to derive the mathematics for dividing space into the above five regions, consider the equation for the ellipsoid, as previously stated,

$$(X-X_c)^T W(X-X_c) = 1$$

and the equation of a straight line through the camera position X_o and the point in question X_p, in parametric form,

$$(X-X_o) = u(X_p-X_o)$$

These can be combined to produce

$$[u(X_p-X_o)+X_o-X_c]^T W[(X_p-X_o)+X_o-X_c] = 1$$

the roots of which determine the intersections of the line and ellipsoid. This equation is equivalent to

$$au^2 + bu + c = 0$$

where

$$a = (X_p-X_o)^T W(X_p-X_o)$$

$$b = 2(X_p-X_o)^T W(X_o-X_c)$$

$$c = (X_o-X_c)^T W(X_o-X_c) - 1$$

The roots of the above equation in u determine the region of space in which X_p lies. If the roots are imaginary ($b^2-4ac < 0$), the point is to the side of the ellipsoid. If the average of the two roots ($-b/2a$) is positive, the point is in front of the midpoint surface, if negative, it is behind the surface. If the roots are real, the point is in front of, behind, or inside the ellipsoid according to whether both roots are greater than unity, both roots are less than unity, or unity lies between the roots, respectively. Alternatively, we can use the fact that the point is outside of the ellipsoid if and only if $(X_p-X_c)^T W(X_p-X_c) > 1$.

The discrepancy of the first kind is

$$\epsilon = \sqrt{(X_p-X_c)^T(X_p-X_c)}\left(1 - \frac{1}{\sqrt{(X_p-X_c)^T W(X_p-X_c)}}\right)$$

In order to compute the discrepancy of the second kind, the midpoint of the intersections of the camera-point line with the ellipsoid is first obtained as follows:

$$X_u = -\frac{b}{2a}(X_p-X_o) + X_o$$

Then the discrepancy of the second kind is

$$\epsilon = \sqrt{(X_u-X_c)^T(X_u-X_c)}\left(1 - \frac{1}{\sqrt{(X_u-X_c)^T W(X_u-X_c)}}\right)$$

Because there may be erroneous points in the data, points which have large discrepancies relative to the size of the ellipsoid are given less weight in the solution. The weighting function used is

$$\omega = \frac{1}{\frac{1 + 2(\sqrt{(X-X_c)^T W(X-X_c)} - 1)^2}{tr(W)} + \sigma^2}$$

where X represents X_p or X_u for discrepancies of the first or second kinds, respectively, and σ is the component of standard deviation of measurement errors in X_p propagated into the discrepancy. (If these are unknown, σ can be zero.) Thus the dimensionless quantity to be minimized (by adjusting X_c and W) is $\Sigma\omega\epsilon^2$, plus some additional terms for *a priori* values yet to be discussed. However, this quantity is minimized only with respect to the effects of X_c and W acting through ϵ and not their effects through ω. In order to solve the above nonlinear problem, the Gauss method [6] is used. This method is equivalent to using the partial derivatives of the discrepancies to approximate the nonlinear problem by a linear statistical model [7], solving the linear problem, and iterating this process until it converges.

On any one iteration the following is done. The current values of X_c and W are used to compute for each point the value of ϵ as define above and the 1-by-9 matrix P, which consists of the partial derivatives of ϵ with respect to the three elements of X_c and the six unique elements of W. (W is symmetrical.) The following summations over all of the points are computed, in which each point in the first category above is not used, each point in the second or third categories appears once, and each point in the fourth category appears twice:

$$H = H_0 + \sum P^T \omega P$$

$$C = C_0 + \sum P^T \omega \epsilon$$

(H_0 and C_0 are used for the *a priori* values yet to be discussed.) Then the 9-by-1 matrix of corrections is

$$D = vH^{-1}C$$

where v is a factor used to improve convergence because of the very nonlinear nature of the problem. (Currently v = 0.5 on early iterations, but v = 1 after a test indicates that this will produce more rapid convergence.) The elements of D are subtracted from the corresponding elements of X_c and W to obtain the improved approximations for the next iteration.

Now the *a priori* values will be discussed. In some cases the points affecting the ellipsoid will be insufficient in number or insufficiently distributed to determine all parameters of the ellipsoid very well. It is therefore desirable to have *a priori* values for some of the parameters with appropriate weight in the solution to constrain them to reasonable default values when the points do not contain sufficient information. When there is ample information in the points, the *a priori* values will have very little effect because of their small weight. The *a priori* values currently used are the ground surface height directly under X_c for the vertical component of X_c, with weight 0.1/tr(M), equality for the diagonal elements of W, with weight $tr(M)^2/10$, and zero for the off-diagonal elements of W, with weight $tr(M)^2/10$, where $M = W^{-1}$. (including tr(M) as shown scales things correctly so that the solution is invariant under a scale factor change.) The effect of the W terms is to try to force the ellipsoid into a spherical shape. These *a priori* terms are put into the solution in the following way. The diagonal element of H_0 corresponding to the vertical component of X_c is 0.1/tr(M), the three diagonal elements corresponding to the off-diagonal elements of W are each $tr(M)^2/10$, and the 3-by-3 submatrix on the diagonal of H_0 in the position corresponding to the diagonal elements of W consists of 2/3 on its main diagonal and -1/3 elsewhere multiplied by $tr(M)^2/10$. All other elements of the 9-by-9 matrix H_0 are zero. Then

$$C_0 = H_0 G$$

where G is a column matrix of the current values of X_c and W, arranged as in D, with the height of the ground directly under the center of the ellipsoid subtracted from the element of G corresponding to the vertical component of X_c. H_0 and C_0 are used in the summations for H and C as previously shown.

5. Breaking and Merging Clusters

Because the preliminary clustering is dependent on local information, it may not produce the best segmentation based on

more global information. Therefore, after ellipsoids have been fit to all of the preliminary clusters, these clusters may be tentatively broken into smaller clusters and merged into larger clusters, new ellipsoids are fit to these clusters by the same process previously described, and a decision on whether to keep or reject each of these actions is made based on the goodness of fit of the ellipsoids to the points.

In order to decide where to break a cluster, for each edge in the portion of the original minimal spanning tree which connects this cluster the quantity $\lambda(1-\rho)$ is computed, where λ is the length of the edge and ρ is the minimum of $\sqrt{(X_p-X_c)^T W(X_p-X_c)}$ for the two points connected by the edge. Then the cluster is tentatively broken at the edge for which this quantity is maximum, of all such edges such that each new cluster formed has at least four points at least one of which has $(X_p-X_c)^T W(X_p-X_c) > 1$ (that is, it is outside the old ellipsoid). This process tends to break the cluster at places furthest inside the ellipsoid, but connecting points that are outside the ellipsoid. If this new clustering is accepted by the criteria described below, the process repeats on the new clusters.

After the above breaking process is finished, any two clusters are tentatively merged if $(X_a-X_c)^T W(X_a-X_c) < 4$ for either cluster, where X_a is X_c for the other cluster, provided that these two clusters were not previously one cluster before breaking. If there is competition for the merging, the cluster pair with the minimum value for this quantity is merged first. If a merger is accepted, further mergers can take place on these clusters by this same process.

The criteria for accepting two clusters or one that resulted from a tentative break or merger are as follows. If $(X_a-X_c)^T W(X_a-X_c) < 1$ for either small cluster, where X_a is X_c for the other small cluster (that is, the center of one ellipsoid is inside the other ellipsoid), the single cluster is chosen. Otherwise, the following quantity is computed for each of the three ellipsoids:

$$q = \frac{\Sigma \omega \epsilon^2}{n-3} + \frac{m}{10}$$

where ϵ and ω are the discrepancies and weights from the last iteration, as defined in the previous section, n is the number of points in the cluster corresponding to this ellipsoid, and m is the number of points below the height threshold but directly above the ellipsoid. If the initial approximation is used as the result, ϵ and ω are obtained from the first iteration, and the first denominator is n instead of n-3. (The second term, containing m, is included to penalize solutions which lie mostly below the ground.) Then the two small clusters are chosen if the sum of their two values of q is less than the value of q for the single cluster. Otherwise, the single cluster is chosen.

6. Results

Fig. 3 shows a stereo pair of pictures taken from the Viking Lander 1 on the surface of Mars. Each picture is 256 pixels by 256 pixels. The pixel spacing is 0.04 degrees in azimuth and elevation. (Thus the field of view is about 10 degrees.) The azimuth and elevation from the left camera to the center of the picture are about 18 degrees and -20 degrees, respectively, relative to the camera positions. The two cameras are 0.8187 meters apart. The distances to the points in the scene range from about 3 meters to about 4.5 meters.

Fig. 4 shows the left picture enlarged, with arrows indicating the points found by the stereo processing. The point of each arrow is at the center of an eight-pixel-square window which was matched to a corresponding area in the other picture. (Thus the resolution of the reduced stereo data is 0.32 degrees in azimuth and elevation.) The arrows are dropped perpendicularly from the point in three-dimensional space computed for this point to a nominally horizontal reference plane 1.5 meters below the cameras, with the base of the arrows on this plane, and are then projected into the picture. Fig. 5 shows the same data as Fig. 4 except that the bases of the arrows rest on a ground plane computed from this data by the ground surface finder. Fig. 6 is the same as Fig. 5 except that it shows only points computed to be at least 5 centimeters above the ground surface.

The 5-centimeter height threshold was used for selecting the points to cluster in the object finder. The minimum distance for breaking the minimal spanning tree to form the initial clusters was also 5 centimeters, and the maximum distance for connecting points was 20 centimeters. (Using zero and infinity for this minimum and maximum produced a sightly different initial clustering but identical final results.)

Figure 3. Stereo pair of Martian surface.

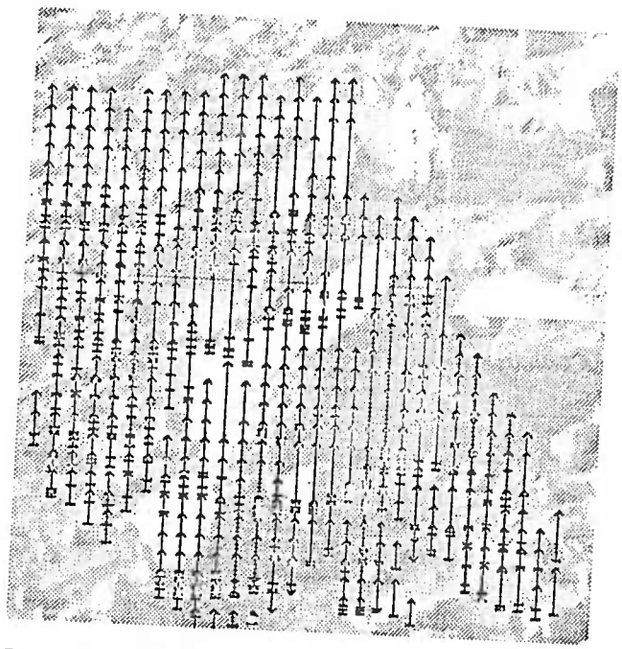


Figure 4. Points found by stereo processing, showing heights above reference plane.

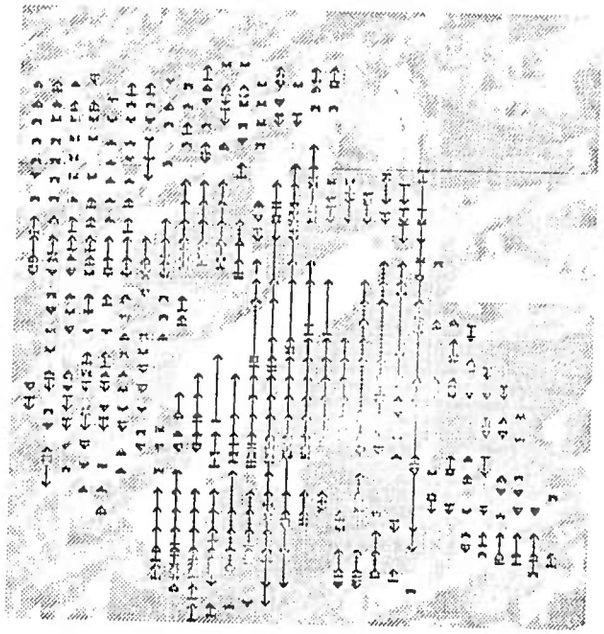


Figure 5. Heights above computed ground plane.

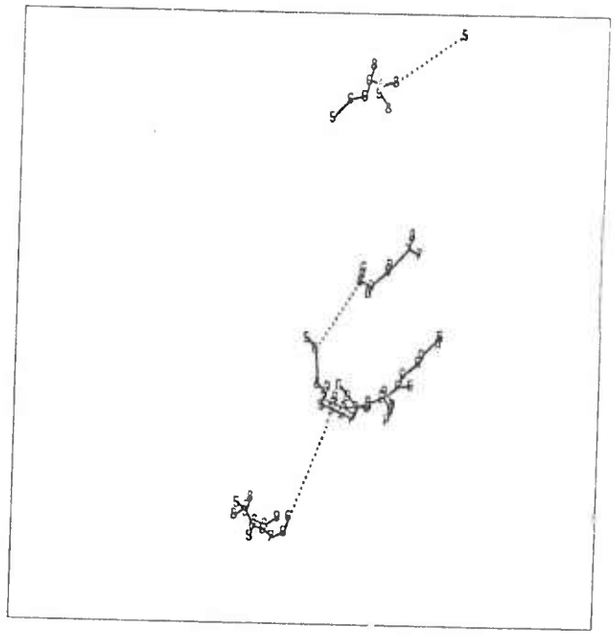


Figure 7. Minimal spanning tree connecting points above 5 cm, vertical view. Solid lines show initial clusters.

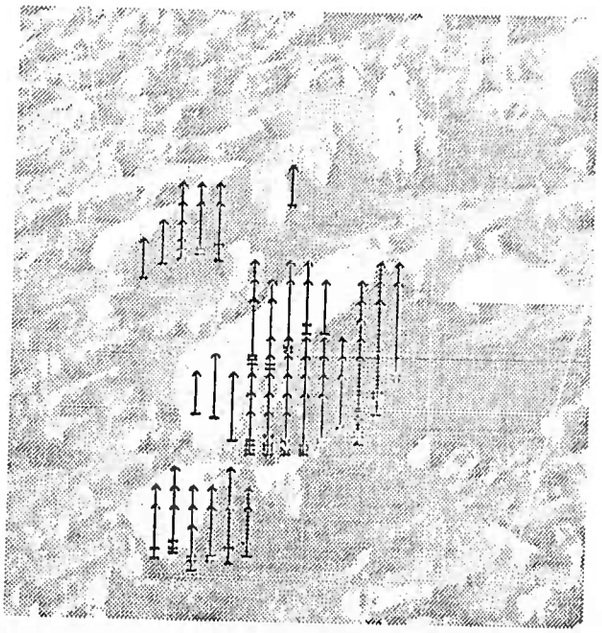


Figure 6. Heights above computed ground plane, for points above 5 cm.

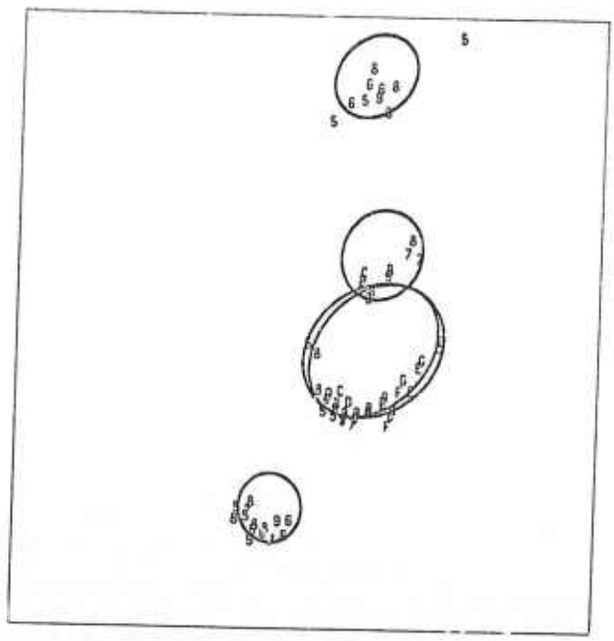


Figure 8. Ellipsoids fit to initial clusters.

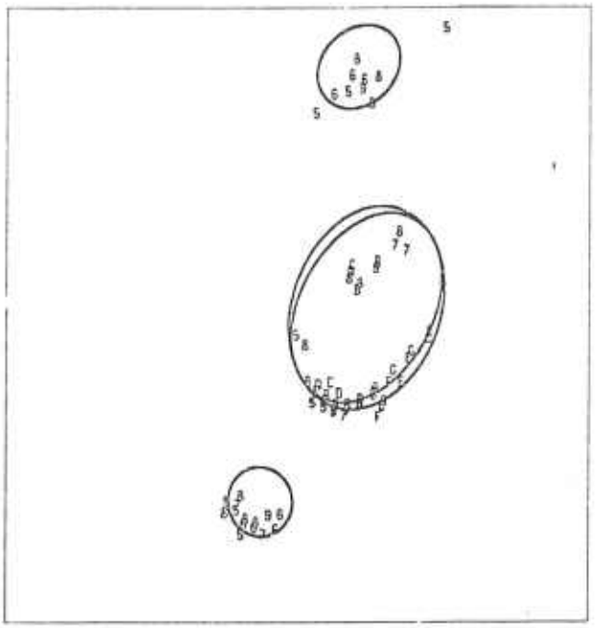


Figure 9. Ellipsoids fit to final clusters.

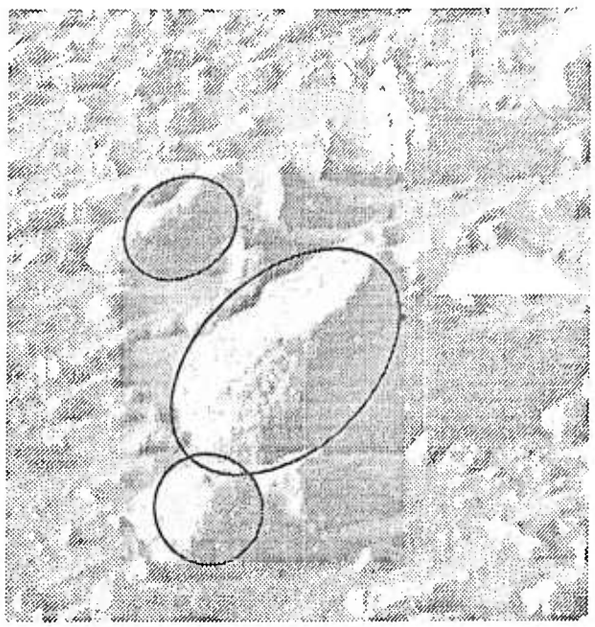


Figure 10. Ellipsoids fit to final clusters, projected into left picture.

Fig. 7 shows the points in Fig. 6 in a nominally vertical orthogonal projection (perpendicular to the reference plane). The figure covers an area one meter by one meter. The symbol for each point represents height in centimeters above the ground plane, with the letters A, B, C, etc. representing the values 10, 11, 12, etc. The points are connected to show the minimal spanning trees that were computed. Solid lines connect points within each initial cluster.

Fig. 8 shows the ellipsoids that were fit to the initial clusters. Each ellipsoid is represented by two ellipses. One ellipse is the orthogonal projection of the ellipsoid onto the reference plane. The other ellipse is the intersection of the ellipsoid with a plane through the center of the ellipsoid and parallel to the reference plane. Only the clustered points are shown here, as in Fig. 7. However, as previously described, any of the points shown in Fig. 4 may have been involved in the adjustment of the ellipsoids. Remember that the fit is done in three dimensions, whereas Fig. 8 shows a two-dimensional projection.

Fig. 9 shows in the same way the results of the breaking and merging operations. The two clusters in the center (corresponding to the large rock in the center of the pictures) were merged into one, and a new ellipsoid is shown for this cluster. The other clusters were not changed.

These results were projected into the left picture to produce Fig. 10. The outline of the ellipsoids as they would be seen from the left camera are superimposed on the picture. The lengths of the principal axes of the large ellipsoid in the center are 36.5, 30.5, and 19.8 centimeters.

Acknowledgments

This work was performed under NASA contract NASW-2918 and the Defense Advanced Research Projects Agency contract MDA903-76-C-0206. The Mars pictures were supplied by Elliott Levinthal and Sidney Liebes of the Viking Lander Imaging Team. This paper was first presented at the Sixth International Joint Conference on Artificial Intellegence, held in Tokyo in August 1979.

References

[1] Gennery, D.B. "A Stereo Vision System for an Autonomous Vehicle." In *Proc. IJCAI-77*. MIT, Cambridge, Mass., August 1977, pp. 576-582. (Also IU Workshop, October 1977.)

[2] Nevatia, R. & Binford, T.O. "Description and Recognition of Curved Objects," *Artif. Intell.* Vol. 8, February 1977, pp. 77-98.

[3] Hohn, F.E. *Elementary Matrix Algebra* (Third Edition). The MacMillan Company, 1973.

[4] Zucker, S.W. "Relaxation Labelling and the Reduction of Local Ambiguities," In *Proc. Third International Joint Conf. on Pattern Recognition*. San Diego, Calif., November 1976, pp. 852-861.

[5] Duda, R. & Hart, P. *Pattern Recognition and Scene Analysis*. Wiley, 1973.

[6] Bard, Y. *Nonlinear Parameter Estimation*. Academic Press, 1974.

[7] Graybill, F.A. *An Introduction to Linear Statistical Models*, Volume I. McGraw-Hill Book Company, 1961.

EDGE BASED STEREO CORRELATION

H. Harlyn Baker

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

An edge based approach to stereo depth measurement is described. Its processing consists of extracting edge descriptions of a pair of images, linking these edges to their nearest (projective) neighbors to obtain the connectivity structure of the images, correlating the edge descriptions in an epipolar [3] reference frame on the basis of local edge properties (here, assuming the input images are registered such that scanlines correspond), and cooperatively removing those edge associations formed by the correlation which violate the connectivity structure of the two images.

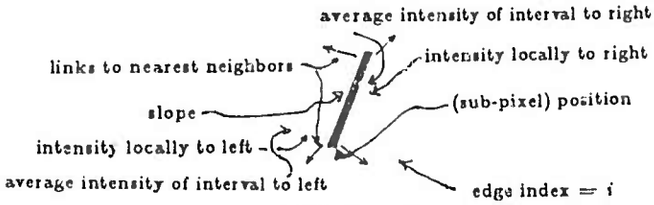
Edge Correlation

The long term interest of this research is *in enabling a computer to build 3-dimensional models of the components of its environment.* To build such models one must have an automatic process for obtaining three-dimensional information from a scene. This is the immediate aim of the work described here - *to develop a vision system capable of obtaining an accurate and reliable edge based depth map of a scene from stereo pairs of views of it.*

Accurate and reliable determinations of the sort needed here require exploitation of the semantic redundancy in the information available to the sensors. The approach to be outlined uses this - intermixing *local* and *global* constraints, constraints derived from observations on the imaging process, in seeking a 3-dimensional interpretation. A correlation procedure chooses the best correspondence of the images using local constraints on a scanline-by-scanline basis, and a cooperative consistency enforcement process works to assure 3-space connectivity using the global constraint of projective connectivity.

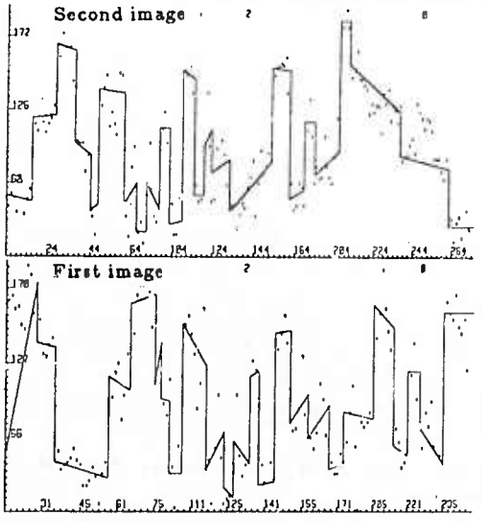
The 3-D correlation was chosen to be *edge* based. This is because of the higher accuracy associated with edge positioning than with area matching, the reduced computation and combinatorics in dealing with edges rather than with area templates, and the desire (at least initially) to work with those parts of the image (and hence of the scene) with the greatest information content - the locus of intensity contrasts between surfaces.

Edges, as they are defined here, occur at those places in an image where the second derivative from a 7×1 or 1×7 bar operator (with lateral inhibition) crosses zero. Each edge has a two dimensional *slope* in the image plane (only those edges with a vertical component in their slope are used in the correlation), a *contrast* across it, average and local *intensity* measures of the intervals to its left and right, 2-D *connectivity* to other edges on prior or subsequent lines, sub-pixel *positioning*, a measure of the extent of its *breadth*, and, increasing across the image, an *index*.



Edge properties
Figure 1

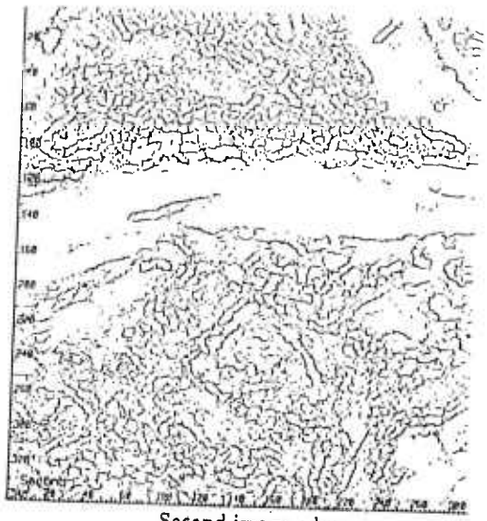
It is these properties of edges along scanlines in the two images which provide the metric for the correlation [1].



First and second image scanline edge depictions
Figure 2

Edge based descriptions of images are also generally more structured than are *area* based ones, as the linking phase of the process establishes edge connectivities in the two-dimensional image. This connection in the image plane, suggesting connectivity in the actual scene, provides the semantic component for a cooperative process to determine the best 3-D interpretation of the scene's edge depiction - the line-by-line edge correlation procedure chooses the best association of first image edges with second image edges from the available local information, then edge connectivity is used to either verify or repudiate these pairings.

Experimentation was done with two basic approaches to the correlation, *contrast* based and *half-edge* based. This report will describe the genesis of the present system as it progressed through these approaches. Accompanying these changes in approach was a change in the basic computation process from a branch&bound search to a Viterbi [2] dynamic programming algorithm. This computation change was brought about by the realization that the analysis of busier images (such as the Night Vision Lab imagery), use of fewer ad hoc'isms in parameter settings (through measures of image statistics for finer control of noise based thresholds), and, as always, the search for greater generality, all lead to increased combinatorics. Although I will begin with a description of *contrast* based branch&bound, much of what is decribed applies to both correlation approaches and both methods of correlation computation.



Second image edges

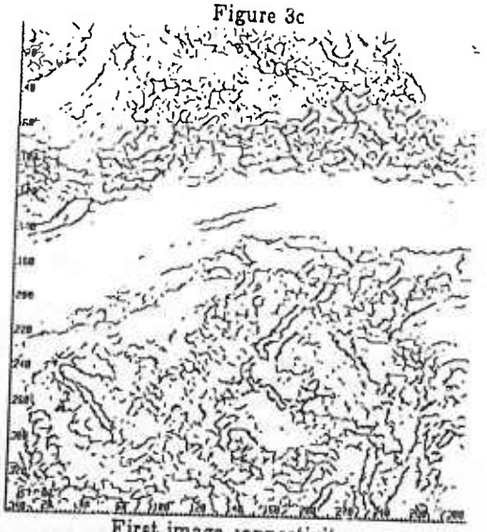
Figure 3a

Second image connectivity

Figure 3b

First image edges

Figure 3c



First image connectivity

Figure 3d

Preparing for the Correlation

Initially, for the branch&bound correlation, edges were grouped by their contrast sign and ordered by their strengths. An edge from the first image would be said to be a *possible mate* to an edge from the second if its *sign* was the same, their *strengths* similar, at least one of the surfaces bounding them to the left or the right being similar enough in *average intensity* in each view to be considered 'the same', and their relative positions *(indices)* being close enough that to associate them together would not exclude too many pairings from the correlation *(implied error)*. The insistence that the edge contrast signs be the same forbids the matching of an edge from say a grey surface projected against a white background with itself from a different view against a black ground (this restriction is not present in the *half-edge* based approach).

A list of such *possible mates* is then formed for each edge of the second image scanline and ordered by a sum of normalized scores of:

(*difference in contrast*)^2,

(*difference in intensity of surfaces to sides*)^2,

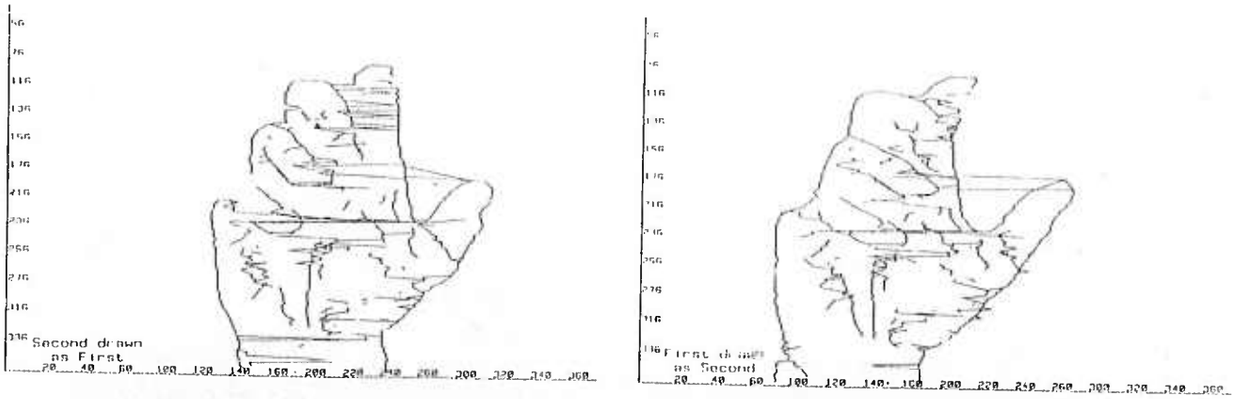
implied error (missed edges, if this association is made),

and a final nonlinear component putting edge matings whose left and right surfaces are *both* matchable to the head of the list. Correlation, then, is the process of finding the 'best' set of possible pairings of the edges.

Branch and Bound Correlation

What drives the correlation is a search for 'explanation' of the greatest number of edges in the scene (an 'explained' edge is one which has been unambiguously positioned in 3-space), and this metric provides the most effective element of the pruning technique (for indeed, the combinatorics can become far too great to allow unbounded expansion of the search tree). A running count of the *implied error* (edges which cannot be correlated) is maintained at each stage of the expansion of the correlation, and any partial assignment giving rise to an *implied error* above that of the 'best-so-far' (initially requiring 50% of edges to potentially match) is denied further expansion at that point.

This first approach, *contrast* based correlation, showed itself to be an acceptable technique with the data used for its development (Figure 4 demonstrates the quality of an early *contrast* based branch&bound correlation, when run with unrealistically high thresholds on a fairly noise-free non-busy stereo pair), but its shortcomings - not allowing contrast reversals to match on one side or another (such as grey moving from a white to a black ground), and requiring the actual step (contrast) to be of similar size - made it necessary to consider other approaches for a more general solution.



This depiction requires some explanation. The figure to the right is produced by following the connectivity in the first image, drawing lines between those edges in that image which have been associated by the correlation process with edges in the second image (ie. they have correlates in the other image), but rather than using the first image's coordinate references, the coordinate reference frame of the second image is used. This means that when following a connected set of edges in the first image, say the back of the hand, everything will look fine as viewed from the other image until an edge associated with something in the second image that's not the back of the hand is encountered. At this point a line will be drawn to whatever part of the second image that edge is associated with, producing a noticeable horizontal jag to that part of the image.

Correlation results
prior to cooperative connectivity enforcement
Figure 4

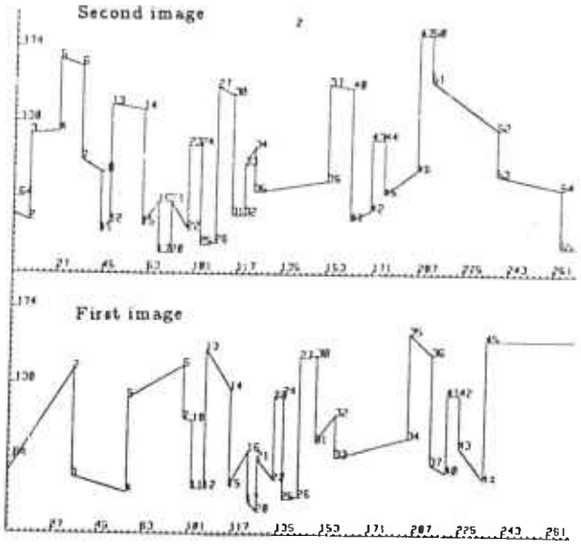
The second approach, *half-edge* based correlation, was developed to allow the possibility of contrast-reversed edges matching on one side or another. Here, the sign and strength of edges are ignored, and only the ordering parameters (as specified above) are used in forming lists of *possible mates* for each second image scanline edge. Unfortunately, there being in effect both a left and a right edge to deal with where before there was only one, the combinatorics tend to get way out of hand (a typical scanline in the Night Vision Lab imagery has about 30 edges, and the size of the search space on some of these was found to be up into the billions).

The combinatoric problem is not confined to *half-edge* based correlation, as in truth it is necessary with both approaches to limit the computation before letting the branch&bound correlation loose. This trimming was done, with both the *contrast* and the *half-edge based* approaches, by removing from consideration those scanline edges (either first or second image) of lowest contrast (where it is presumed that edges of lower contrast are less significant to the modelling) until the total estimable combinatorics passed under some prespecified maximum (50000 permutations). (*Implied error* bounding rarely let the actual number of permutations required exceed the hundreds).

Now, with either of the two approaches, the scanline correspondence having greatest number of 'explained' edges (in case of equivalent counts, lowest sum of squared intensity difference at sides of edges) is chosen as the result of the correlation. This is a set of pairs (see Figure 5):

$$\{(f_i, s_j) \mid f_{\frac{i}{2}} \textit{ correlates with } s_{\frac{j}{2}};$$
$$i, j \textit{ even} \rightarrow \textit{left of edge};$$
$$i, j \textit{ odd} \rightarrow \textit{right of edge}\}$$

Clearly there will be miscorrelations among these - experience, and a little thought, have shown that they can surely be expected to occur near the periphery of the scanline, where the need to correlate bounding edges is not present (the global constraint of maximising the edge pairings has diminishing effect near the sides, where the relative displacement of the images means no correlation exists). And of course there will always be edges that do look alike. With this local ambiguity, one can expect to always have incorrectly assigned edge pairs. It is up to a further analysis with more global information to remove these miscorrelations.

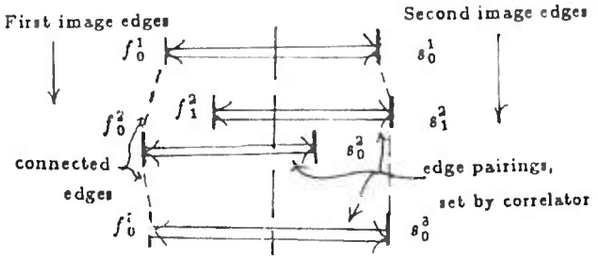


{(2, 4) (5, 5) (6, 6) (7, 7) (10, 10) (11, 11) (12, 12) (13, 13) (14, 14)
(15, 15) (16, 16) (17, 17) (21, 21) (22, 22) (23, 23) (24, 24) (25, 25)
(26, 26) (27, 27) (30, 30) (35, 33) (36, 34) (37, 35) (41, 37) (42, 40)
(43, 41) (44, 42) (45, 43) (46, 44)}

Set of associated pairs (as Figure 2)
f-first image, *s*-second image
Figure 5

Cooperative Continuity Enforcement

To the rescue comes a depth continuity enforcement process operating in a cooperative mode upon the edge pairings assigned by the correlation. It follows connected edges in the two image planes, removing those edge pairings that it finds to be inconsistent. An inconsistent pairing, in this sense, is one whose edges are nearest connected in 2-space (as seen in either image) to edges which have been paired differently by the correlation. This conflict in correlation is a necessary condition for inconsistency, but is not alone sufficient. For each pairing (f_i^m, s_j^m) on scanline m, and associated disparity δ_{ij}^m, measures η and σ are kept of the mean and standard deviation of changes in δ_{ij} among 2-space connected pairings. The other half of the consistency criterion is that the change in disparity $|\delta_{ij}^m - \delta_{pq}^n|$ from the pairing $(f_i^m, s_j^m)(disparity = \delta_{ij}^m)$ to its nearest connected 2-space neighbour pair $(f_\nu^r, s_q^n)(disparity = \delta_{pq}^n)$ be within $[\eta - \sigma, \eta + \sigma]$. A single such conflict is not enough to remove a pairing (as, really, which pairing is in error?). Rather, a pairing is only removed when it is found to be inconsistent from 2 different sources.



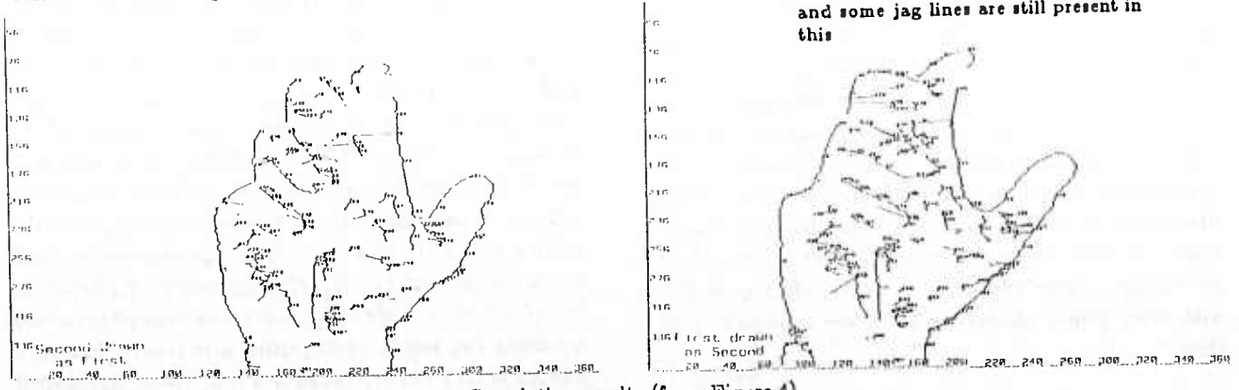
connectivity without consistent correspondence
Figure 6

When a correlation pairing (f_i^m, s_j^m) is removed, no immediate attempt is made to reassign the f_i^m or s_j^m (although it will be in the near future), and these edges are bypassed in the 2-space connectivity structure .. the paired edges above and below them are joined now as 2-space nearest connected pairs. The new change in disparity is evaluated, and tested to see whether it lies in the $[\eta - \sigma, \eta + \sigma]$ interval. When no further edge pairings can be removed by this process, all pairings left having a single inconsistency are removed. This more ruthless removal could not be applied earlier, as it would delete good pairings adjacent to bad ones in the process of removing the bad .

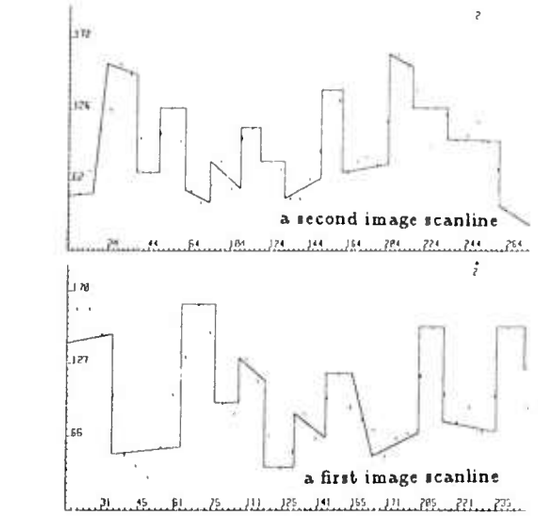
Certainly, a great deal more may be done with these edges unpaired by the removal process ... 2-D connectivity may make it clear where they should be really paired, the reduced combinatorics (generally, as fewer edges are left unpaired than began that way) may facilitate further correlation with relaxed constraints (particularly, allowing edge reversals - a left-right ordering in one image matching a right-left ordering in the other), etc. ... effort will be put in this direction later, once the correlation process itself is felt to be sufficiently stable and successful.

Coarse to Fine Correlation

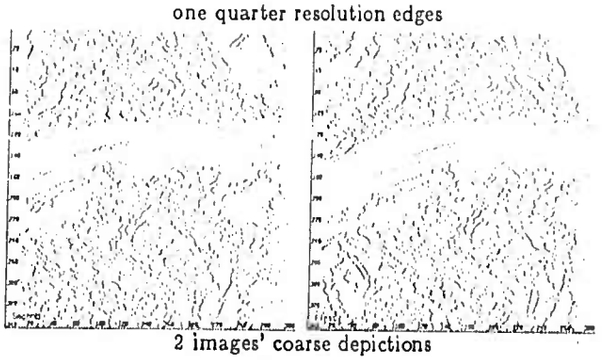
A concern over the loss of potentially useful edge correlations through the combinatorial reduction then lead to a further change in the approach. In it *contrast* based and *half-edge* based correlation approaches are combined. It uses a resolution reduced 'planning' scheme that exploits the coarse image structure in reducing the correlation combinatorics. Here, the images are repeatedly halved in resolution with a 1-2-2-1 averaging operator (in effect removing the high frequency components, leaving the low-frequency, coarser structure more visible). The edges found at this resolution in the first and second images are treated as significant, or *landmark* edges, and are correlated in the *contrast* based edge style. For each set of assignments of these edges having *implied error* not greater than that of the 'best-so-far', the edges in the intervals between the *landmarks* are *half-edge* correlated. The reasons for this dual mode of operation are that: 1) it seemed a nice way to merge the two (equally valid and equally incomplete) edge matching assumptions, and 2) it was felt that the smoothed image edges would be more contrast preserving over viewpoint, having their high frequencies removed (this is a valid assumption for narrow to medium angle stereo, perhaps not for wide angle).



Correlation results (from Figure 4)
after cooperative connectivity enforcement
Figure 7



one quarter resolution edges



2 images' coarse depictions

Figure 8

There is always the danger with hierarchically constrained searches such as this that the best solution at a coarse level (reduced resolution) will not be consistent with the best overall solution at the finest level (full resolution). This problem was avoided in the branch&bound correlation by doing the full resolution interval correlation whenever the reduced resolution correlation suggested an improvement was possible over the best yet achieved. This recursive approach cannot be integrated into the Viterbi correlation as it stands, and this is one aspect of the Viterbi algorithm that we are looking into. Viterbi does, however, have some pretty outstanding advantages: it is optimal (when using the same assumption of no edge reversals in the images), polynomial (as opposed to exponential), and is very very fast.

Some recent results with the Viterbi algorithm follow. The principal complaints with these at the moment are the large number of apparently poor correlations (jag lines)they have, and the lack of good vertical connectivity in the image edge depictions. Our efforts to solve these problems are leading us to consider adding other local constraints for the edge matching, keeping alternative selections for each edge pairing, and examining forms of interpolatory correlation [4] in the surviving intervals defined by the correlation and cooperative consistency enforcement process. We have confidence that these will significantly improve the results. We would also like to work with more controlled images (remember the requirement that the epipolar lines here be parallel to the camera baseline, only in the CDC synthetic images was this the case), and our recent reinstallation of a pair of GE CCD cameras will help in this.

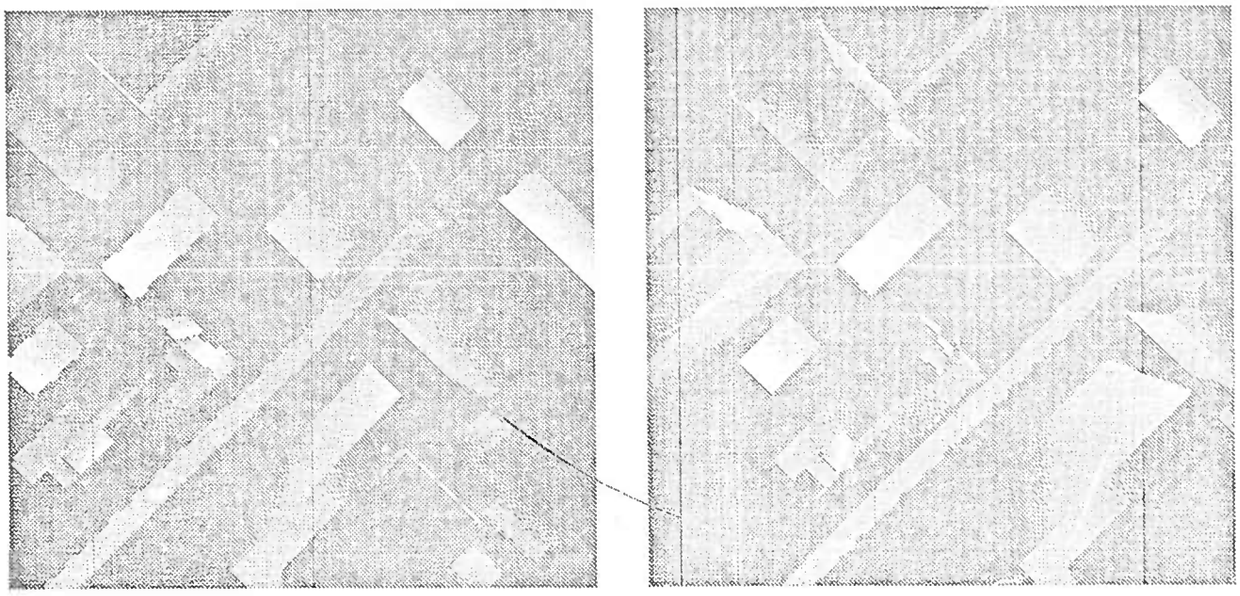
References

[1] Arnold, R. David , 'Local Context in Matching Edges for Stereo Vision,' *Proc. ARPA Image Understanding Workshop*, Cambridge, Mass. May 1978, 65-72.

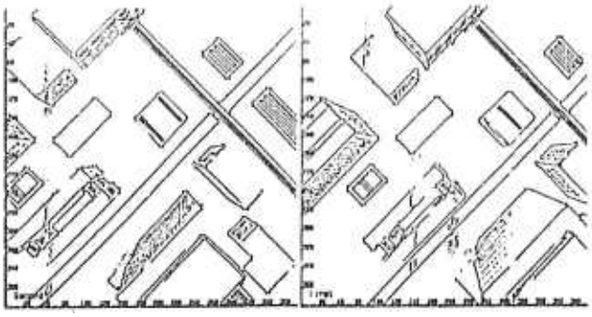
[2] Forney, G. David Jr., 'The Viterbi Algorithm,' *Proc. IEEE*, Vol. 61, No. 3, March 1973.

[3] Henderson, Robert L., Walter J. Miller, C. B. Grosch, 'Automatic Stereo Recognition of Man-Made Targets,' *Soc. Photo-Optical Instrumentation Engineers*, Vol. 186, August 1979.

[4] Panton, Dale J., 'A Flexible Approach to Digital Stereo Mapping,' *Photogrammetric Engineering and Remote Sensing*, Vol. 44, No. 12, December 1978, 1499-1512.

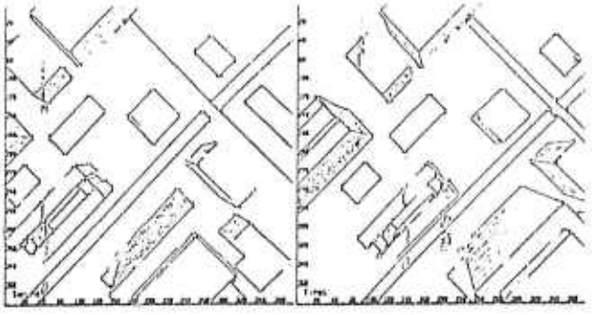


Synthesized urban images (from CDC) with some vertical glitches



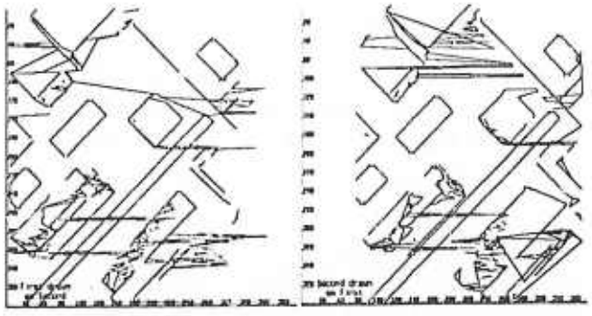
All edges

including laterally inhibited edges



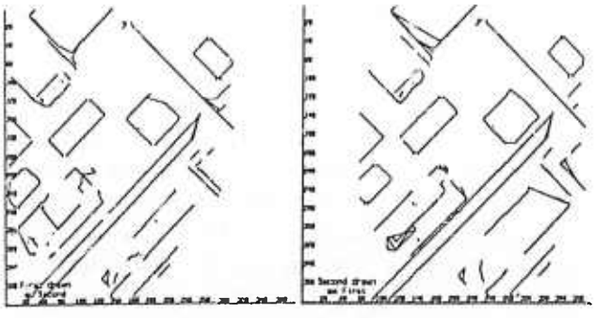
Excluding laterally inhibited edges

laterally inhibited edges are kept, but not correlated



After correlation

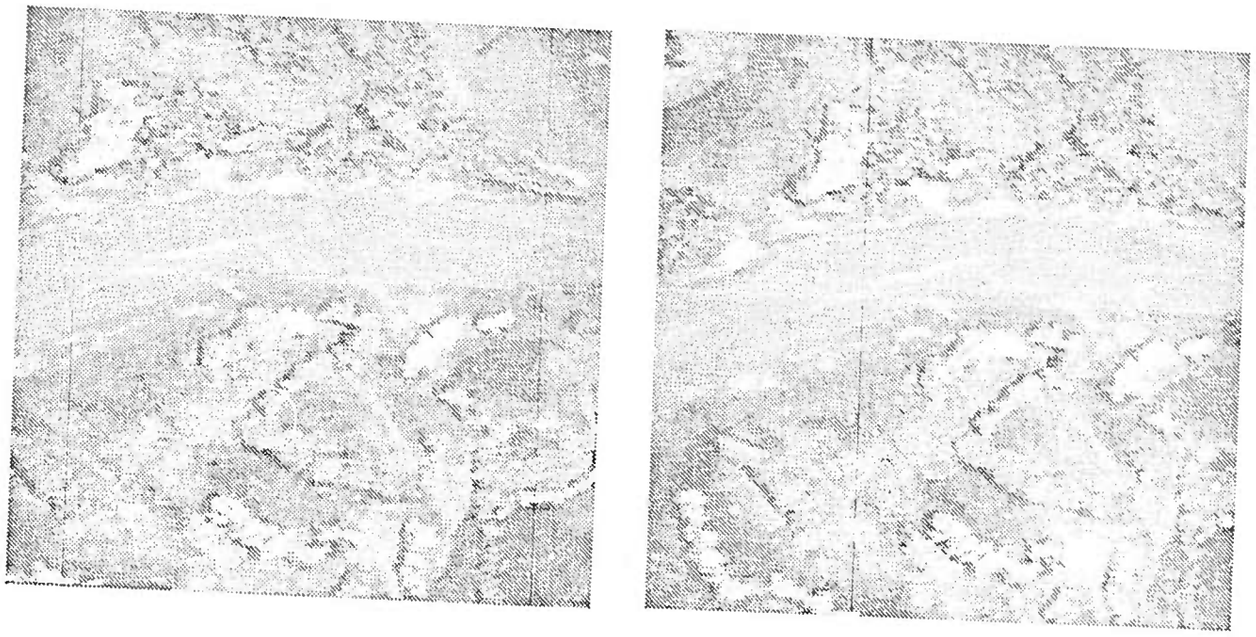
Left side of edge correlations are superimposed with right side of edge correlations



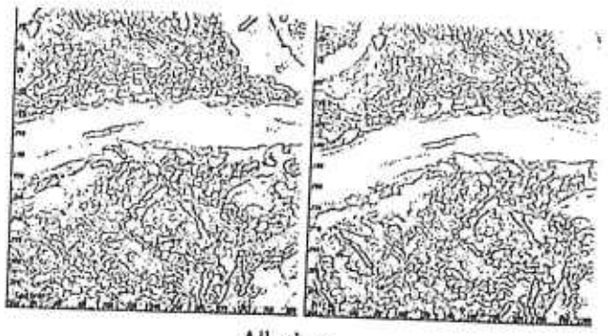
After cooperative process

3900 half-edge pairings (57%of possible edges) 7 sec. first image, 12 sec. second image plus correlation. 37%removed by cooperative process.

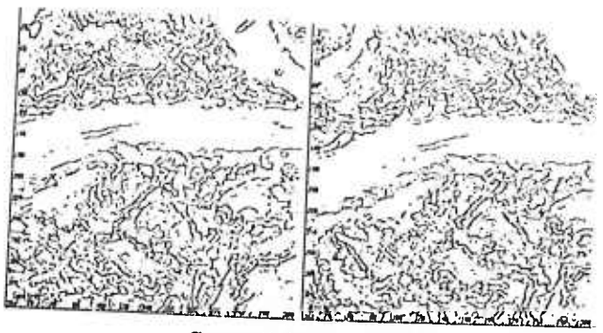
Figure 9



Night Vision Lab imagery

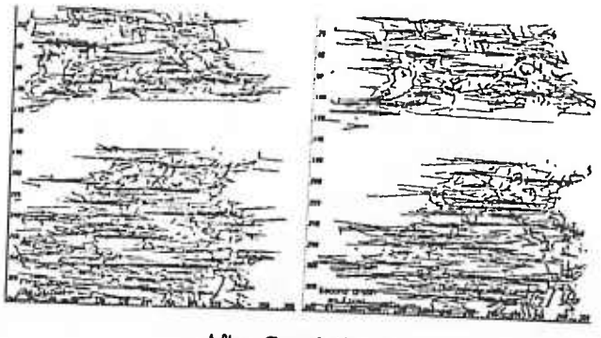


All edges

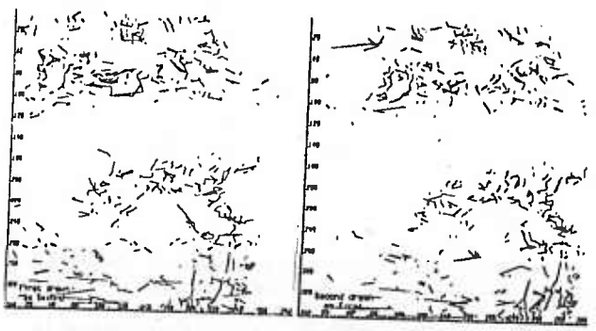


Connectivity structure

The lack of good vertical continuity here handicaps the cooperative process greatly. The image is almost a texture.



After Correlation



After cooperative process

5300 half-edge pairings (65%of possible edges) 5 sec. first image, 15 sec. second image plus correlation. 48%removed by cooperative process.

Figure 10

CONVERGENCE PROPERTIES OF TWO LABEL RELAXATION

Arden R. Helland

Westinghouse Systems Development Division
Baltimore, Maryland 21203

ABSTRACT

This paper analyzes the convergence properties of the Peleg relaxation scheme [1] for the one-dimensional, two-label case. It is shown that if a label probability is identically zero, then that label probability will remain zero. However, if non-zero, then the label probability will either increase toward unity or decrease toward zero, depending on whether the average probability of the neighborhood is above or below a threshold determined by the relationship of the compatibility coefficients. It is shown that the speed at which ambiguity is resolved is reduced by the unlike compatibility coefficients. Speed can be increased with no change in the final results by reducing the unlike coefficients to zero.

INTRODUCTION

Relaxation labeling has been shown to provide reduced error rates in pixel classification compared to procedures that are based solely on local evidence. Relaxation classification uses initial label probabilities, which may be modified based on the label probabilities in the neighborhood (region) of each decision point (pixel).[1,2] The objective for the cases of interest is to segment darker regions from lighter regions in the presence of significant noise which causes ambiguity within the regions. The process appears to have the intriguing capability of making a very faint target in selected windows appear obvious after a few iterations.[3,4] It, therefore, appears that relaxation simultaneously provides the advantages of classification by regional (vs local) evidence as well as the constrast of segmentation, but using the principal of deferred commitment to resolve ambiguity.

THRESHOLD

The following analysis is based on the probability updating rule developed by Peleg for two labels. The two labels will be called b and w, so that Pib is the probability that the ith pixel is black. Likewise, Pjb is the probability that the jth neighbor is black. In the general case, the compatibility coefficients are a function of the orientation of the jth neighbor to the ith pixel, as well as their labels. This paper is based on the assumption that the compatibility coefficients are independent of (or averaged over) the orientation. Therefore it is a function only of the labels, so that for labels α and β:

$$rij\ (\alpha,\beta) \rightarrow r\alpha\beta$$

For labels b (black) and w (white), therefore, we have rbb, rbw, rww, and rwb. rbb and rww are called the alike label coefficients; rbw and rwb are the unlike label coefficients.

The updating rule uses the following intermediate summations where K is the iteration index:

$$Sb^K = Pib^K \sum_{j=1}^{N} Pjb^K \cdot rbb + Pib^K \sum_{j=1}^{N} Pjw^K \cdot rbw$$

$$Sw^K = Piw^K \sum_{j=1}^{N} Pjw^K \cdot rww + Piw^K \sum_{j=1}^{N} Pjb^K \cdot rwb$$

These are used to generate probabilities for the next iteration as follows:

$$Pib^{K+1} = \frac{Sb^K}{Sb^K + Sw^K} \quad \text{and} \quad Piw^{K+1} = \frac{Sw^K}{Sb^K + Sw^K}$$

Sb^K and Sw^K may be reduced to two variables by making the following substitutions:

$$\hat{P}\alpha = \frac{1}{N} \sum_{j=1}^{N} pj\alpha^K$$

$$\hat{P}w = 1 - \hat{P}b$$

Also, for simplicity, the iteration index, K, will be dropped; where K+1 occurs it will be indicated by *. Therefore, by multiplying by the number of neighbors N and combining terms, we have

$$Sb = Pib\ [\hat{P}b(rbb-rbw) + rbw] \equiv Pib \cdot Fb$$

$$Sw = [1-Pib]\ [(1-\hat{P}b)(rww-rwb)+rwb]$$

$$\equiv [1-Pib] \cdot Fw$$

Now the change in the black probability of the ith pixel for each iteration is defined as follows:

$$\Delta Pib \equiv Pib^* - Pib = \frac{Sb}{Sb + Sw} - Pib$$

$$= \frac{Sb - Pib\ (Sb+Sw)}{Sb + Sw}$$

$$= \frac{Pib\ (Fb-Fw)\ (1-Pib)}{Pib \cdot Fb + (1-Pib)\ Fw}$$

The denominator must be non-zero for all normal cases; the only condition that would permit both Sb and Sw to be zero is if Pib and $\hat{P}b$ have absolutely certain but opposite labels.

Because the product Pib (1-Pib) is a quadratic function, it is zero for either Pib = 0 or Pib = 1, and positive for all other values $0 < Pib < 1$, with a maximum value occurring at Pib = .5. Therefore, ΔPib depends on the Fb - Fw term as follows:

$$Fb - Fw \geq 0 \Rightarrow \Delta Pib \geq 0$$

Now, Fb and Fw are dependent only on $\hat{P}b$ and the compatibility coefficients, so that $Fb - Fw \geq 0$ yields the following result:

$$\hat{P}b \geq \frac{rww - rbw}{rww - rbw + rbb - rwb} \equiv Tb$$

Therefore, if the average black probability of the neighborhood ($\hat{P}b$) is above the threshold Tb, the black probability of the ith center pixel will be increased. If $\hat{P}b$ is identically equal to Tb, Pib is unchanged. Of course, Tw = 1 - Tb.

Therefore, we may conclude the following regarding the response of ΔPib to Fib and $\hat{P}b$.

1. For lighter regions:
   $Pib < 1$ and $\hat{P}b < Tb \Rightarrow \overset{*}{Pib} < Pib$
2. For darker regions:
   $Pib > 0$ and $\hat{P}b > Tb \Rightarrow \overset{*}{Pib} > Pib$
3. If $\hat{P}b < Tb$ for all iterations, $\overset{*}{Pib} \rightarrow 0$
4. If $\hat{P}b > Tb$ for all iterations, $\overset{*}{Pib} \rightarrow 1$

Therefore, Pib = 0 and Pib = 1 are values toward which all pixels within consistent regions will converge. Although $\hat{P}b$ may change from one iteration to the next, it is unlikely to change its relationship to the threshold (except if it is very near the threshold or if the initial label probabilities are inconsistent).

SPEED

Inspection of Fb and Fw shows that the unlike coefficients rbw and rwb have a tendency to reduce ΔPib.

Now, 1-Pib is the difference between Pib and absolute black label probability. Therefore, relative speed is defined as the ratio of ΔPib to the remaining difference; this cancels the 1-Pib term in the numerator.

$$\text{Therefore, } Cb = \frac{\Delta Pib}{1 - Pib} = \frac{Pib\ (Fb-Fw)}{Pib \cdot Fb + (1-Pib)Fw}$$

It can be shown that Cb is maximum with respect to Pib and $\hat{P}b$ when both approach the constraint limit of unity. Therefore, if Cb is evaluated at Pib = $\hat{P}b$ = 1:

$$Cb = \frac{Fb - Fw}{Fb}$$

$$= \frac{rbb - rwb}{rbb}$$

$$\text{Likewise, } Cw = \frac{-\Delta Pib}{Pib} = \frac{Fw - Fb}{Fw} = \frac{rww - rbw}{rww}$$

These expressions show that non-zero unlike coefficients directly reduce speed of convergence.

RESULTS

The preceding analysis has shown that the speed of convergence toward absolute labels and the point of divergence threshold may be derived from the compatibility coefficients. These measures of speed and threshold were computed for some of the cases of coefficients used by Rosenfeld and Smith in their paper "Thresholding Using Relaxation", in the publication by the University of Maryland Computer Vision Laboratory.[4] The principal cases for the "dark tank picture" are reproduced here as Figures 1 through 5 (they were originally Figures 10, 6, 13, 11 and 12). The computed measures of speed and threshold are tabulated as follows:

Figure No.		1	2	3	4	5
Speed	Cd	.28	.38	.29	.33	.89
Threshold	Tl	.77	.74	.58	.50	.50

The first four figures are given in order of decreasing light threshold (increasing threshold for the dark object). Therefore, the size of the object should tend to decrease as less of its fuzzy boundary is classified as dark. This is fairly readily observable in the representation for both the updated probabilities (displayed as revised gray level) and the histogram display for each iteration. Ideally, the threshold region should develop a "valley" in the histogram as values converge to the two extremes. Although this tendency does occur, the following reasons make this less obvious:

1. Movement near the threshold region is slow, so that separation on initial iterations is slow until ambiguity is resolved.
2. Unless the light and dark thresholds are both 0.50, the region with lower threshold expands into its fuzzy boundary region. For light thresholds greater than .5 (figures 1 through 3) the dark object expands, so there is continued motion from right to left through the threshold region of the histogram.
3. The histograms are normalized to the highest value, so convergence to the most common label tends to suppress the remaining portion of the histogram.

The first four figures show varying degrees of speed, all relatively slow. Figure 2 has the highest speed measure; examination shows that segmentation of the image is more complete (and the histogram more completely converged to the two extremes) than for Figures 1, 3 or 4.

Figures 4 and 5 have the same threshold, but offer a dramatic illustration of the difference in speed of convergence. Examination shows that iteration 8 of Figure 4 indicates results approximately comparable to iteration 2 of Figure 5. The faster convergence of Figure 5 obtains virtually complete segmentation. In addition, because the light and dark thresholds are equal, the boundary of the object is stationary.

This is also indicated by the convergence of the histogram to the two extremes, with no change after segmentation is complete.

Figure 5 did not achieve segmentation of the entire object because the light threshold was so low that the less dark regions of the object were laheled "light". Rosenfeld and Smith define a process whereby the image gray values may be transformed to adjust the probabilities so that the desired gray value results in the probability threshold. Of course, selection of the desired gray level theshold depends on the type of imagery and the application of the image processing results.

CONCLUSIONS

Boundary stability is related to the relaxation thresholds. If the threshold for one of the labels is less than for others, then regions with that label will tend to expand. However, small regions tend to shrink due to the effect of the surroundings on convex boundaries. This appears to imply that for large regions of interest, most shape detail will be preserved with (nearly) equal thresholds for each label. However, the response to small regions of one label may be improved by a moderate reduction in that threshold. The expansion due to unequal thresholds may be used to counteract the tendency for small regions to shrink.

The analysis of speed showed that the change in probability of the center pixel is reduced by the proportion of the unlike coefficient to the alike coefficient for each label. Maximum speed is obtained when unlike coefficients are zero. Also, it is shown that if the probability for the center pixel is zero for either label, then that probability will not change as a result of further relaxation iterations. Therefore, input data should avoid zero probability to permit full operation of the relaxation functions.

When the average probability of the neighborhood is equal to (or nearly equal to) the threshold, then the probability of the center pixel is unchanged (or changed only slightly). If gray levels near the probability threshold are considered as ambiguous levels, then little change occurs until the ambiguity becomes resolved by further iterations. In other words, the segmentation process inherent in relaxation proceeds very cautiously as long as the neighborhood labeling remains ambiguous, deferring commitment to any label until information from more distant regions resolves the ambiguity. However, if the neighhorhood labeling is not near the probability threshold, then the label probabilities are driven toward zero and unity at a rate that is maximized if the unlike coefficients are zero.

REFERENCES

1. S. Peleg, A New Probalistic Relaxation Scheme TR-711, Computer Science Center, University of Maryland, November 1978.

2. A. Rosenfeld, R.A. Hummel & S.W. Zucker, Scene Labeling by Relaxation Operations, IEEETSMC-6

3. A. Rosenfeld, A. Danker & C. Dyer, Blob Extraction by Relaxation, Proceedings: Image Understanding Workshop, April 1979, 61-65.

4. A. Rosenfeld & R. Smith, Thresholding by Relaxation, Computer Science Center, University of Maryland.

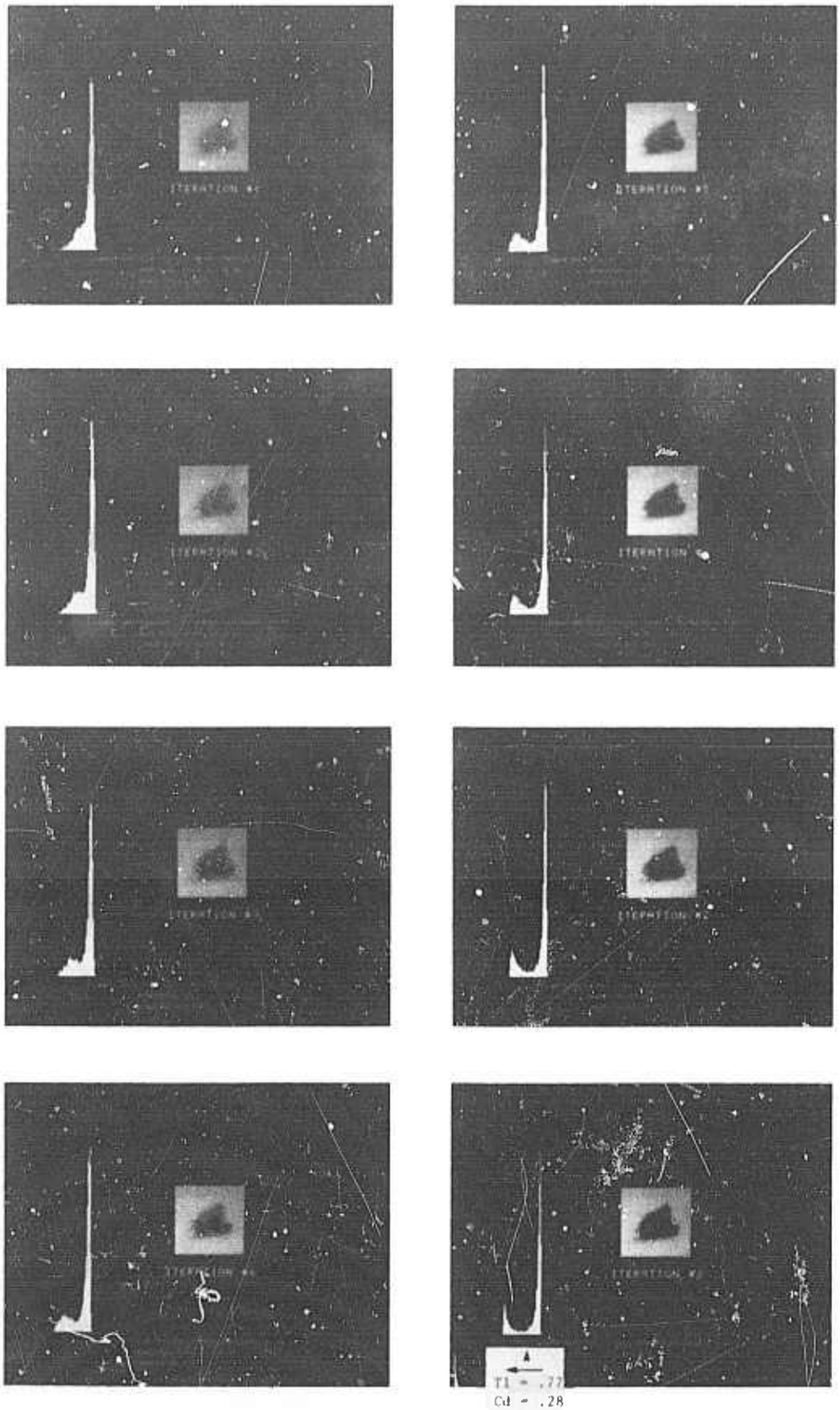
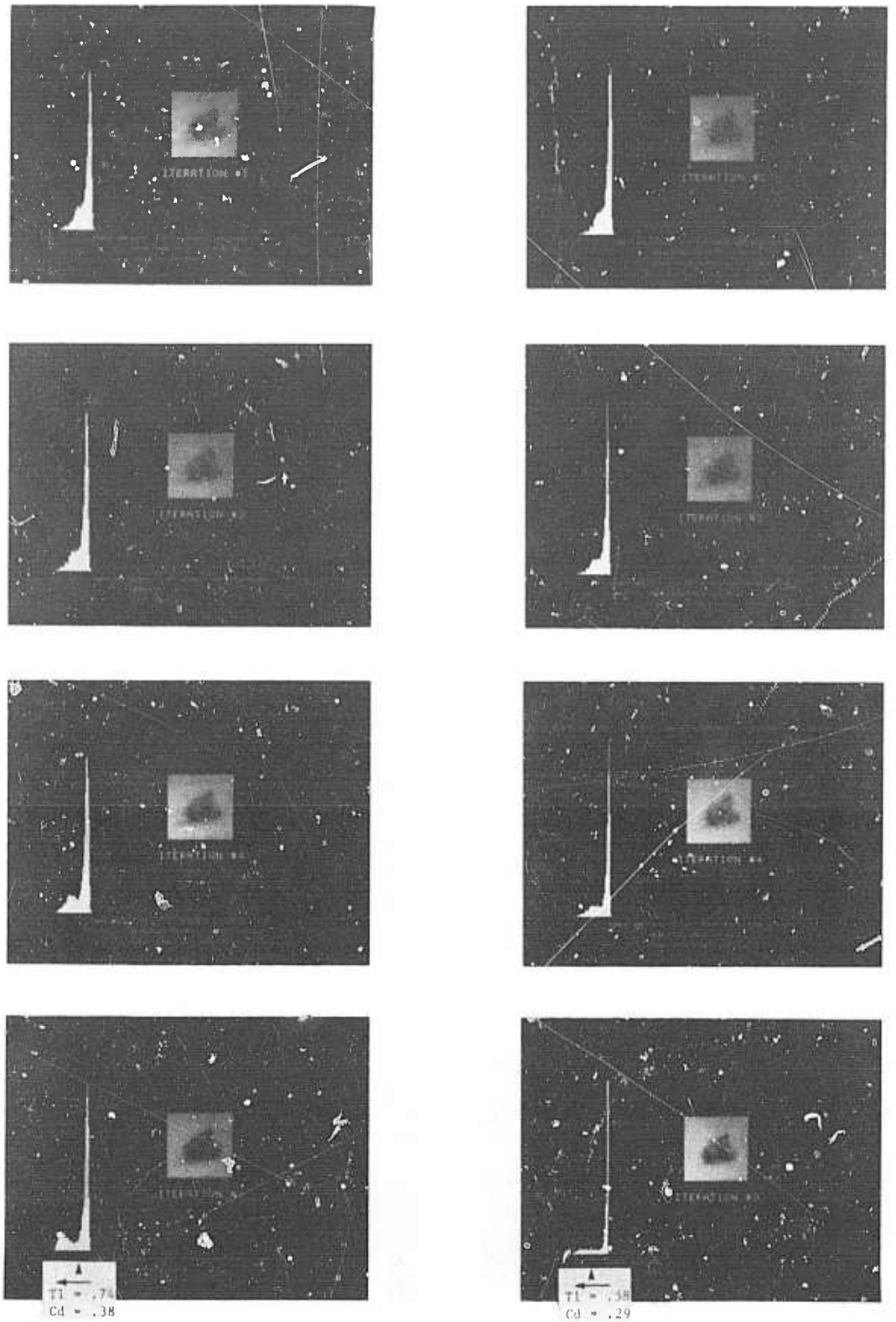
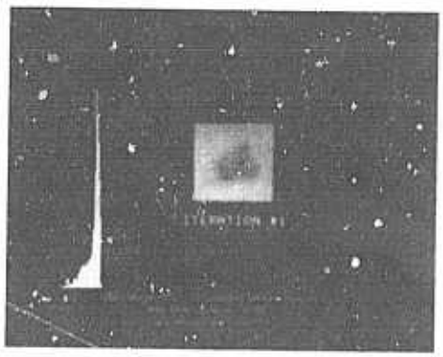
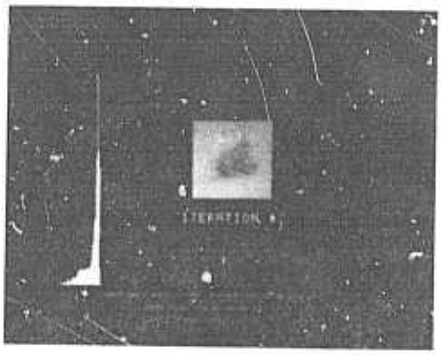
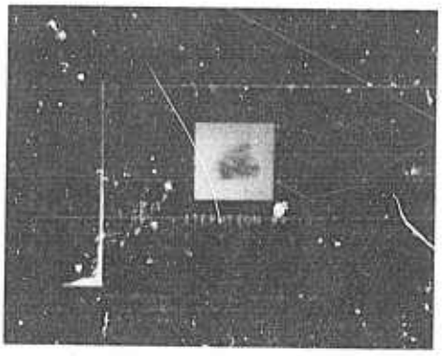

ITERATION
ITERATION
ITERATION
ITERATION
ITERATION
ITERATION
ITERATION
ITERATION
TI = .77
CD = .28

Figure 1


Tl = .74
Cd = .38
Tl = .58
Cd = .29







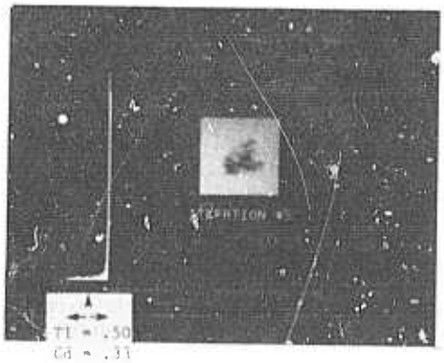
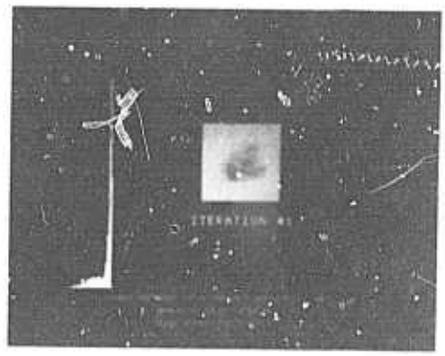
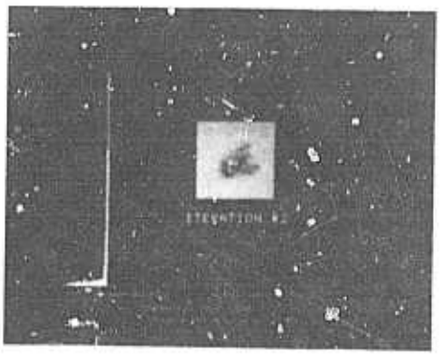
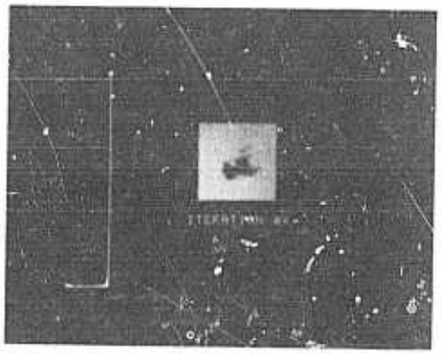


Figure 4







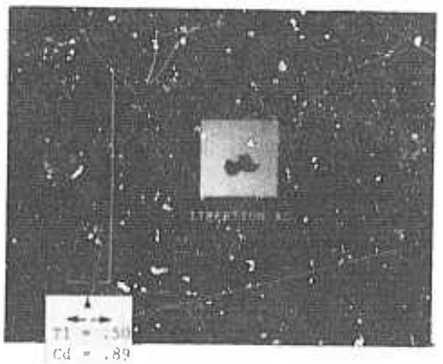


Figure 5

INVESTIGATION OF VLSI TECHNOLOGIES FOR IMAGE PROCESSING

WILLIAM L. EVERSOLE AND DALE J. MAYER*

TEXAS INSTRUMENTS INCORPORATED
P.O. Box 222013, M/S 3407
Dallas, Texas 75222

ABSTRACT

This paper summarizes recent work performed by Texas Instruments Incorporated for Carnegie-Mellon University on the investigation of very large scale (VLSI) implementations for image processing. To fully exploit VLSI technologies, system engineers must appreciate the constraints as well as the benefits of having a million transistors on a single chip. Optimization of the architecture of an image processor is imperative to ensure a broadly applicable programmable component. Discussion of the architecture trade-offs and the hardware implementation of a programmable image processor to implement a sum of products operator is presented.

INTRODUCTION

To take advantage of rapid advances in integrated circuit technology, the hardware architecture of an image processor must be optimized in terms of execution time, size, power, and cost. This optimized architecture will allow the implementation of highly complex image processing functions on a monolithic substrate. Examples of the need for optimized architectures are image processing algorithms that involve array multiplications of the form

$$y = \sum_{i=0}^{I-1} a_i x_i \qquad (1)$$

where the a_i's represent a set of fixed weighting coefficients known a priori and the x_i's represent a set or a sequence of input values. Equation (1) can be used to calculate the coefficients of various transforms such as Fourier, Cosine, Hadamard, Haar, and others used in many signal processing systems. Where two dimensional transforms are needed, successive one dimensional transforms can be used if the transforms are separable.

Many image processing algorithms require the discrete convolution of a two dimensional input image with a two dimensional convolution array. For these algorithms, Equation (1) can be represented in two dimensions by

$$y(m_1,m_2) = \sum_{n_1}^{N_1} \sum_{n_2}^{N_2} x(n_1,n_2)A(m_1-n_1+1,m_2-n_2+1) \qquad (2)$$

where $y(m_1,m_2)$ represents the discrete convolution of the input $N_1 \times N_2$ image array, x, and the convolution array, A. These mathematical operations are often based on neighboring pixel values and are termed neighborhood operators. Examples of neighborhood operators include noise smoothing in which the components of A are of low pass form, edge crispening in which the components of A are of high pass form and linear edge enhancement in which several masks are convolved with the original image to produce maximum output for different slope directions.

Equation (1) can be implemented using digital multipliers and adders, however the size and power required to perform the multiplications at video data rates with the accuracy needed for most image processing applications is prohibited from a hardware standpoint. A technique for realizing equation (1) that does not require digital multiplication is the distributed arithmetic technique.[1-3] Distributed arithmetic allows the implementation of a sliding sum of products (convolution) of an input word sequence with a set of weighting coefficients using a table look-up procedure. Distributed arithmetic techniques can also be used to implement a nonsliding sum of products of an input word set with a set of weighting coefficients.

This paper discusses distributed arithmetic and the hardware implementation of a programmable sum of products operator based on distributed arithmetic.

DISTRIBUTED ARITHMETIC

The operation performed by the distributed arithmetic technique is the convolution defined

---

*Mr. Mayer is now with Northwest Datacom

by

$$y_n = \sum_{i=0}^{I-1} a_i \, x_{n-i} \qquad (3)$$

where x_{n-i} is a B-bit word represented by

$$x_{n-i} = \sum_{j=0}^{B-1} x_{(n-i)j} \, 2^j \qquad (4)$$

$$x_{(n-i)j} \in \{0,1\}$$

Substituting Equation (4) into Equation (3) yields

$$y_n = \sum_{j=0}^{B-1} \sum_{i=0}^{I-1} a_i \, x_{(n-i)j} \, 2^j \qquad (5)$$

Since the values a_i are fixed, the 2^I possible values of the bracketed term of Equation (5) may be calculated a priori and stored in a memory. For each j of the outer summation, the value of the bracketed term is recalled from the memory location whose address is formed by the I bit binary word

$$Z = [x_{(n)j}, x_{(n-1)j}, x_{(n-2)j}, \cdots x_{(n-I+1)j}] \qquad (6)$$

The word stored in this location is given by

$$C(Z) = \sum_{i=0}^{I-1} a_i \, Z_i \qquad Z_i \in \{0,1\} \qquad (7)$$

These values recalled from memory are weighted by the factor 2^j and summed over j.

Tradeoffs between memory size, number of adders and the number of table look-ups were discussed at a previous workshop.[4]

ARCHITECTURES

For many image processing applications such as video bandwidth reduction, forward-looking infrared (FLIR) autocueing, target classification, and image understanding, algorithms based on Equation (1) are used which operate on fixed 8 by 1 blocks or sliding 3 by 3 blocks of pixels. Real-time video data rates of 10 mega-samples/second and 6- to 8-bit words are desired.

The distributed arithmetic technique can be used to implement these algorithms with both fixed and sliding blocks of data. Figure 1 shows a block diagram of a distributed arithmetic implementation capable of operating on 9 by 1 or 3 by 3 blocks of pixels. The 6-bit input words (A, B, and C) are sampled at 10 MHz and can be loaded into three addressable latches as three words in parallel or as three

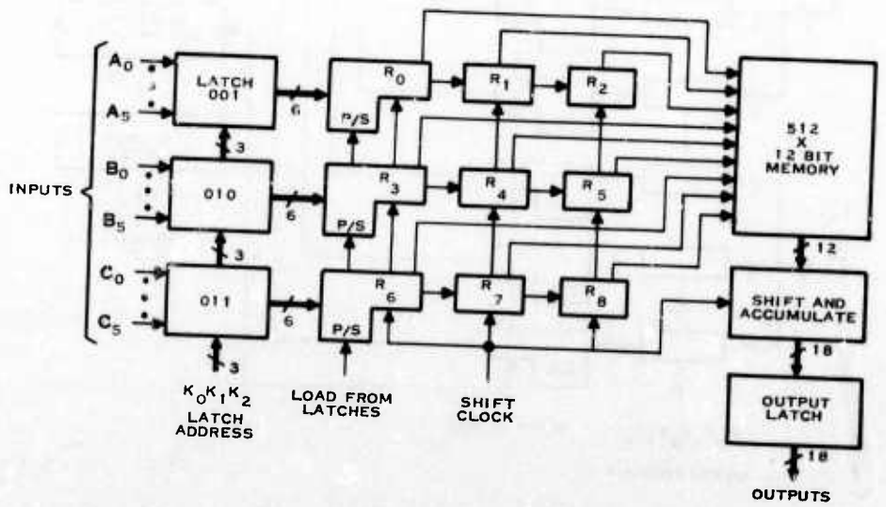


Figure 1. Distributed Arithmetic Implementation of a Programmable Sum of Products Operator

words sequentially. After all three latches are loaded, the parallel-to-serial shift registers (P/S) convert each bit-parallel input word into a bit-serial word which is clocked sequentially through three six-stage shift registers. The outputs of the final stage of each shift register form the 9-bit memory address.

When operating on 9 by 1 pixel blocks, the three inputs (A, B, and C) are connected together. The three input latches are loaded sequentially and the parallel-to-serial registers are loaded every third sample period; therefore, the shift registers, memory, and accumulator must operate at twice the sample rate, i.e., 20 MHz.

When operating on sliding 3 by 3 pixel blocks, the three input latches are loaded in parallel at the sample rate and, therefore, the 6-bit shift registers, memory, and accumulator must operate at six times the sample rate, i.e., 60 MHz.

With maximum memory blocking and multiple-bit addressing this frequency can be reduced to 10 MHz at the expense of eight adders. By using pipelining techniques, each addition can have a maximum of 100 ns to produce a result.

An alternative approach is the ROM-accumulator (RAC) technique which sacrifices the ability to operate on sliding blocks of data for a reduction in the internal clock rate. A block diagram of a RAC implementation is shown in Figure 2. Nine addressable latches demultiplex the input data. The 6-bit input words (A, B, and C) are sampled at 10 MHz and are loaded into the latches as three sets of three parallel words or as nine sequential words. The parallel-to-serial registers convert each input word to bit-serial form for addressing of the memory.

When operating on 9 by 1 pixel blocks, the three inputs (A, B, and C) are connected together and nine sample periods are required to load the nine addressable latches.

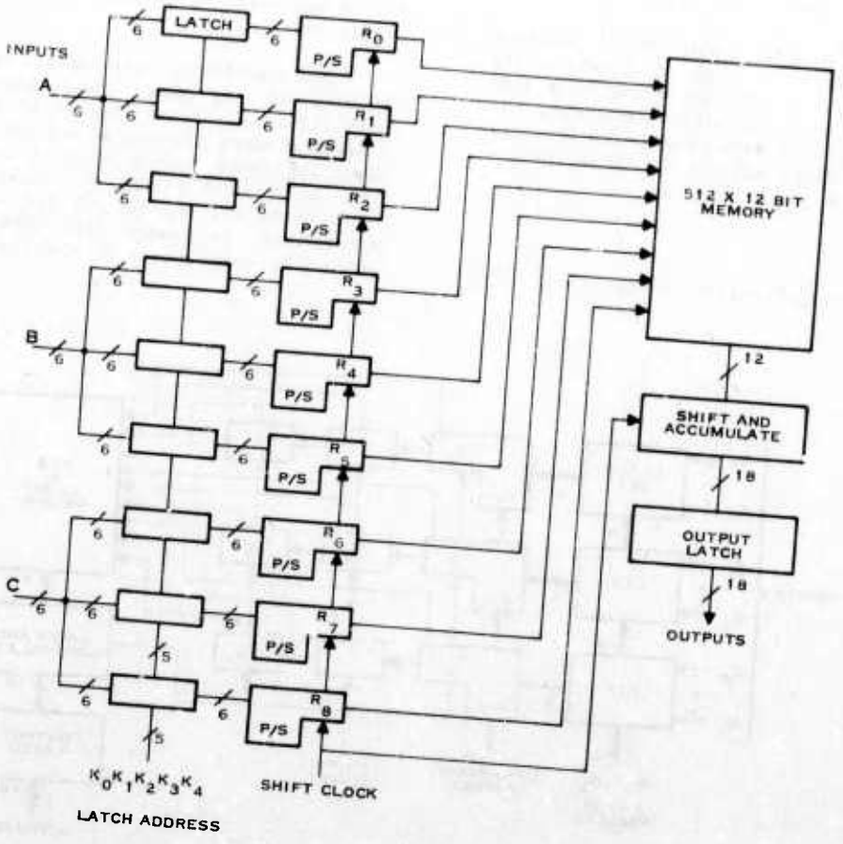


Figure 2. RAC Implementation of a Programmable Sum of Products Operator

When operating on 3 by 3 pixel blocks, three sample periods are required to load the nine addressable latches. Assuming continuous operation, this requires the parallel-to-serial registers to address the memory six times in 300 ns, i.e., 20-MHz internal clock rate. Pipelining techniques can be used to give the parallel-to-serial registers, memory, and accumulator each a maximum of 50 ns to settle.

A second RAC approach which reduces the internal clock rte by a factor of two is shown in Figure 3. Multiple-bit addressing reduces the required internal clock rate to 10 MHz, and memory blocking reduces the memory size by a factor of three. Two addres must be included to sum the three memory outputs. With this architecture, pipeline techniques would allow 100-ns settling time for each element.

A third RAC approach, shown in Figure 4, reduces the internal clock rate by a factor of six over that of Figure 2. Multiple-bit addressing reduces the required internal clock rate to 3.33 MHz, and memory blocking keeps the memory size to about the same as that of Figure 2. Eight adders must be included to sum the nine memory outputs. Pipeline techniques would allow 300-ns settling time for each element of Figure 4.

Although a sliding sum of products is not directly performed with a RAC implementation, three RAC processors can be operated with three-phase clocking to perform convolution. Thus, a RAC equivalent of a distributed arithmetic processor is three times larger. However, the required internal clock rate is reduced by a factor of three.

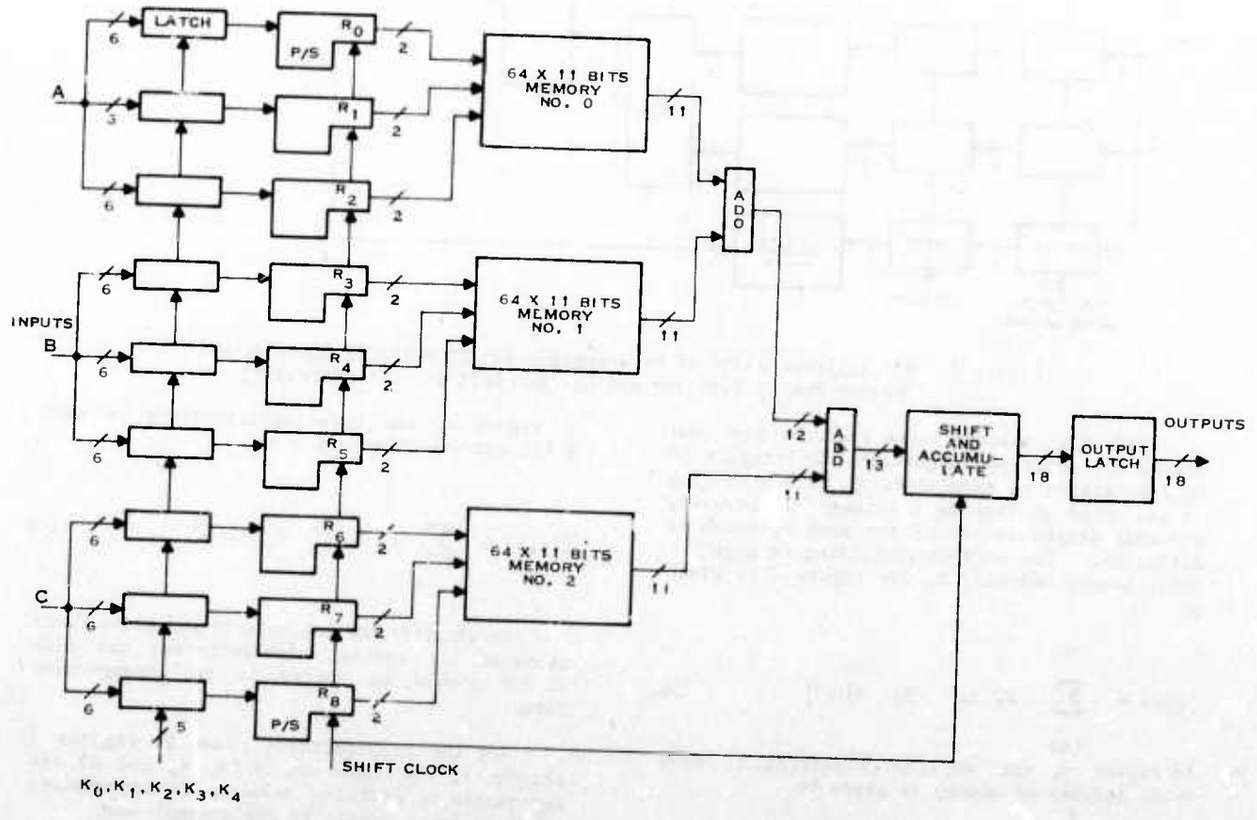


Figure 3. RAC Implementation of Programmable Sum of Products Operator With Memory Blocking and Multiple-Bit Addressing

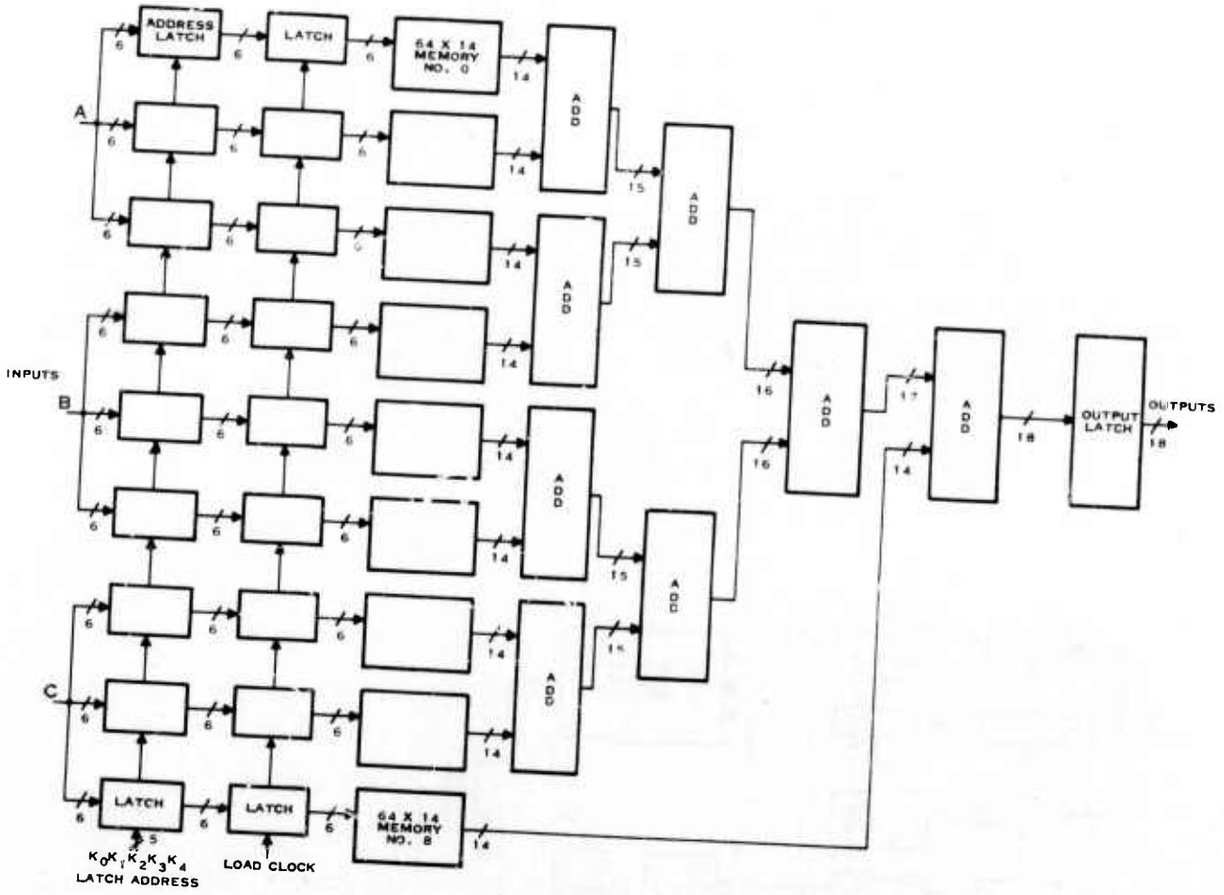


Figure 4. RAC Implementation of Programmable Sum of Products Operator With Maximum Memory Blocking and Maximum Multiple-Bit Addressing

Each RAC implementation produces the same output for positive (magnitude only) inputs if full precision is maintained at each operation as indicated in Figures 2 through 4. However, the code stored in memory for each approach is different. The word corresponding to each 9-bit memory address, Z, for Figure 2 is given by

$$C(Z) = \sum_{i=0}^{8} a_i Z_i \qquad Z_i \in \{0,1\} \tag{8}$$

In Figure 3, the word corresponding to each 6-bit address of memory is given by

$$C(q,Z) = \sum_{i=0}^{2} a_{3q+i}[Z_{2i}+2Z_{2i+1}] \tag{9}$$

$$Z_{2i}, Z_{2i+1} \in \{0,1\}$$

In Figure 4, the word corresponding to each 6-bit address of memory q is

$$C(q,Z) = \sum_{m=0}^{5} a_q Z_m 2^m \qquad Z_m \in \{0,1\} \tag{10}$$

To simplify the accumulator and adder functions of the various architectures, the code in ROM should be stored in two's-complement form.

For the architectures shown in Figures 1 through 3, the input words (A, B, and C) are restricted to positive values. To accommodate negative input values in two's-complement form, a subtraction option in the accumulator function is required for Figures 1 and 2. The memory output from the sign bit address (i.e., the 9-bit address obtained from the sign bits in registers R_0-R_8) is subtracted in the

accumulator. For input value representation other than two's complement, the architecture complexity is further increased. Figure 3 cannot readily accommodate negative inputs due to the multiple-bit addressing. i.e., the sign bits are not distinguishable from the other bits.

The architecture of Figure 4 accepts negative inputs using any representation, i.e., two's complement, sign magnitude, offset binary, etc. This is due to the fact that with maximum memory blocking and maximum multiple-bit addressing, the architecture of Figure 4 looks up each complete product term, $a_i x_i$, of Equation 1, rather than a partial product. Equation 10 must be modified to accommodate the chosen number representation.

Another advantage of the architecture of Figure 4 is that the function stored in each memory may be any linear or nonlinear function of the input signal, x_i, i.e., the values stored are not limited to the linear product of x_i with a weighting coefficient a_i. For example, the values stored in each memory may represent the square of x_i and, thus the architecture of Figure 4 implements the sum of squares. Other nonlinear functions of the x_i's; such as polynomials, trigonometric functions, exponential functions, magnitude functions, etc., may be implemented in each memory. This allows nonlinear masking opeations, such as logarithmic Laplacian edge detection to be performed.

HARDWARE IMPLEMENTATION

To fully explore the architectural implications of a programmable sum of products operator, a hardware implementation of Equation (1) has been designed and fabricated. The breadboard is capable of operating on sliding blocks of data as illustrated in Figure 1 and fixed blocks of data as illustrated in Figure 2. A block diagram of the hardware implementation is shown in Figure 5. The breadboard consists of nine input latches, nine parallel in-serial out shift registers, a fast 512x12 bit memory for temporary storage of the partial products, an EPROM for permanent storage of the partial products, shift and accumulate circuitry, tri-state output latches, and control circuitry.

The input atch structure is hardware or software selectable for serial data entry at DATA INPUT C or parallel data entry at DATA INPUT A, DATA INPUT B, and DATA INPUT C. This facilitates the implementation of a 9 point transversal filter or a 3 x 3 sliding window operator, respectively. The input data word length is hardware selectable from 1 bit to 8 bits, and is hardware or software selectable as two's complement or magnitude format.

The weighting coefficients, a, determine a set of partial products which are stored in a 512 x 12 bit high speed random access memory. The data to be stored in each memory location is given by Equation (7). These partial products may be down loaded on the data bus from an external source by a controlling processor. The partial products are hardware selectable as two's complement or magnitude format. The partial products obtained from the RAM during operation are summed in a carry-save accumulator with data shifting to weight the significance of each partial product. Output data (20 bits) is buffered by a tristate output latch.

The data input section consists of 9 latches connected to form a 9 stage long by 8 bit wide shift register. The outputs of each stage are also connected to 9 parallel to serial conversion registers which form the data for the RAC operation. The data input operation is controlled by the INPUT CONTROLLER. The controller has a hardware or software selected BCD value count as one of its inputs. The INPUT RESET line is pulsed low to initialize the controller. The INPUT STROBE line shifts data through the input latches and clocks the controller at its leading edge. After a sufficient number of strobe pulses occur, the INPUT CONTROLLER generates a LOAD pulse which latches the data from the input latches into the parallel to serial registers and starts the high speed asynchronous RAC CONTROLLER. The LOAD pulse also resets the INPUT CONTROLLER so that new data can be shifted into the input latches while the RAC operation is taking place. A second INPUT RESET pulse is not required.

The RAC CONTROLLER operates from an asynchronous internal 16.7 MHz oscillator (60 nsec period). This is the maximum clock rate for the components selected for the shift registers and accumulator. After a LOAD pulse is received from the INPUT CONTROLLER, the RAC CONTROLLER sequences the operations of the PARALLEL to SERIAL REGISTERS, the PARTIAL PRODUCT MEMORY, the SHIFT AND ACCUMULATE function, and the OUTPUT LATCH. It also provides signals which may be monitored by an external processor if desired. After the final INPUT STROBE pulse, the result of the RAC calculation is available at the TRI-STATE OUTPUT LATCH in $[B_x + 6\ 1/2]$ internal clock cycles. The additional 6 1/2 clock cycles is due to the pipelined architecture.

Thus,

$$t_{RAC} = [60 \times B_x + 390]\text{nsec} \qquad (11)$$

where t_{RAC} is the time required to process the data and Bx is the number of significant bits in each input data word. For 8 bit data, this is 870 nsec. The RAC CONTROLLER will bring the READY line high when the output is available and will not allow another result to overwrite the output latch data until the READ ACK input is pulled high by the controlling processor. The OUTPUT BUFFER FULL line goes high, the INPUT STROBE line is inhibited, and the internal oscillator is stopped if an over-

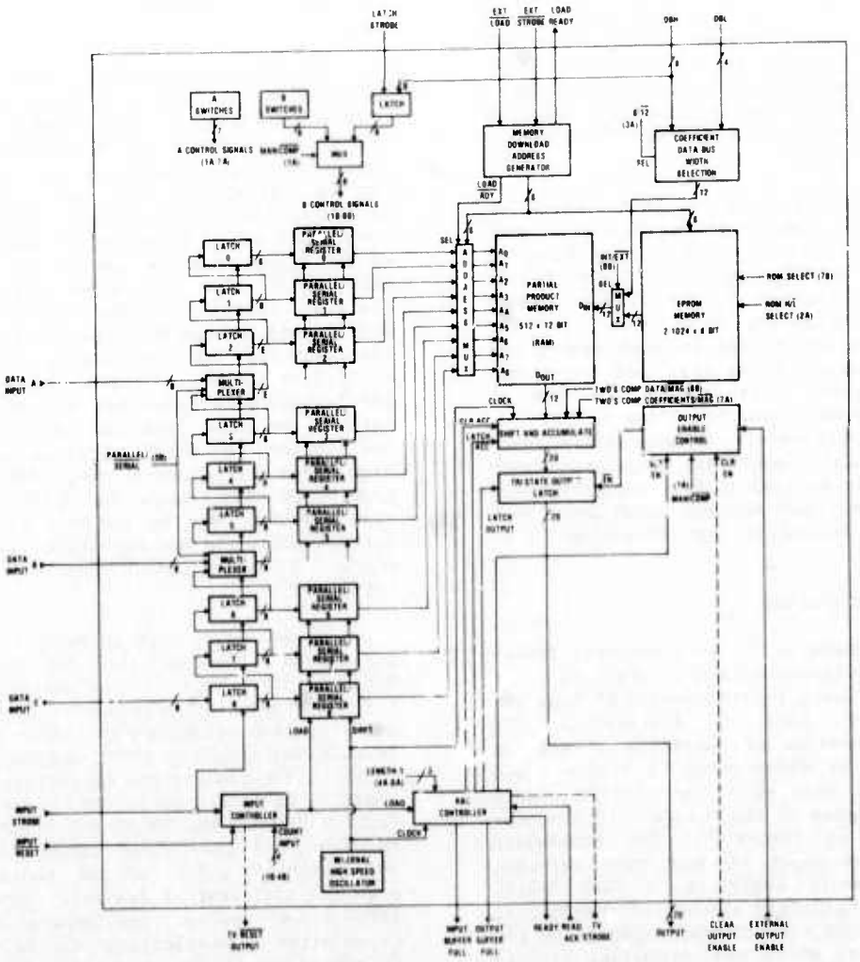


Figure 5. Programmable Sum of Products Unit

write condition exists. The tristate outputs are enabled by pulling the external OUTPUT ENABLE line low.

Due to the pipeline organization of the RAC hardware, the PARALLEL/SERIAL REGISTERS may be loaded as soon as B_x bits of data have been shifted out of them, i.e., B_x internal clock cycles of 60 x B_x nanoseconds after the last INPUT STROBE pulse. For 8 bit data, this is 480 nsec which gives a throughput rate of about 2 MHz. If the INPUT CONTROLLER generates a LOAD pulse before the PARALLEL/SERIAL REGISTERS are emptied, the INPUT BUFFER FULL line will go high and the INPUT STROBE line will be inhibited until the registers are emptied in order to prevent overwriting any data.

From the above discussion it can be seen that the maximum input data rate depends on the number of bits in the input data words (B_x) and the number of INPUT STROBE cycles. The number of INPUT STROBE pulses (N) between parallel to serial conversions determine the type of operations performed. For sliding 3x3 or 9x1 filter applications only one strobe pulse is needed between parallel to serial conversions. For non sliding 3x3 window operation three strobe pulses are needed and for an 9x1 transform, nine strobe pulses are needed. The maximum input data rate is given by

$$f_{MAX_{IN}} = (N)/(B_x \times 60 \text{ nsec}) \quad (12)$$

For sliding window or transversal filter applications with 8 bit data; f_{MAXIN} = 2 MHz. For applications such as an 8 x 1 transform with 8 bit data; f_{MAXIN} = 16 MHz.

The maximum output data rate is always given by

$$f_{MAX_{OUT}} = 1/(Bx \times 60 \text{ nsec}) \quad (13)$$

The breadboard also has the capability to operate in parallel with other breadboards in order to provide throughput at real time (TV) data rates. Three control lines shown dashed in Figure 5 are provided to synchronize this operation. These lines, TV RESET OUTPUT, TV STROBE, and CLEAR OUTPUT ENABLE are not normally connected when operating a single breadboard.

The breadboard is 12" x 12" x 5", weighs 7lbs and dissipates 15 watts.

CONCLUSIONS

Architectures for performing array multiplication without multipliers have been discussed. Tradeoffs between memory size, processor speed and flexibility were made and a detailed discussion of a hardware implementation of a ROM accumulator processor was presented. This hardware consisted of standard off the shelf components requiring more power and size than an integrated circuit version, however the breadboard is an invaluable aid in the definition of a LSI/VLSI implementation. The breadboard version allows evaluation of the algorithm as well as the discovery and evaluation of the problems, risks and options.

Recent advances in component technology now make possible the realization of digital processors capable of performing array multiplications, however an understanding of architectural tradeoffs that may affect algorithm and hardware performance is necessary before actual integrated circuit designs begins to prevent a poorly defined function.

Under a parallel contract to the Air Force Avionics Laboratory at Wright-Patterson Air Force Base, Texas Instruments is designing a programmable image processing element based on distributed arithmetic using n-channel metal oxide semiconductor technology.

REFERENCES

1 Burrus, C.S. "Digital Filter Structures Described By Distributed Arithmetic," IEEE Transactions on Circuits and Systems, Vol, CAS-24, (December 1977) p 674.

2 DeMan, H.J., Vandenbulcke, C.J., and Van Cappellen, M.M., "High Speed NMOS Circuits for ROM-Accumulator and Multiplier Type Digital Filters," IEEE Journal Solid-State Circuits, Vol. SC-13, (October 1978) p 565.

3 Classen, T.A.C., Mecklenbrauker, W.F.G., and Peek, J.B.H., "Some Considerations on the Implementation of Digital Systems for Signal Processing," Phillips Research Reports, Vol. 30, (1975) p 73.

4 Eversole, W.L., Mayer, D.J., Frazee, F.B., and Cheek, T.F., "Investigation of VLSI Technologies for Image Processing", Proc. IMAGE UNDERSTANDING WORKSHOP, Menlo Park, California, April 1979, p 159.

APPLICATION OF LSI AND VLSI TO IMAGE UNDERSTANDING ARCHITECTURES

S.D. Fouse, G.R. Nudd, and V.S. Wong

Hughes Research Laboratories
3011 Malibu Canyon Road, Malibu, California 90265

ABSTRACT

We describe here the work undertaken at Hughes Research Laboratories (Malibu, California) in support of the DARPA Image Understanding (IU) program. This report covers the period from October 1979 through April 1980 and, as such, represents a transition period for us. Our work prior to this period was concerned with the investigation and demonstration of large-scale integration (LSI) microelectronic technology for image understanding. The principal aims of this work were the design, fabrication, and demonstration of high-speed primitives for real-time processing with low-level operators. This work is continuing in that we are completing a performance evaluation of the 14 low-level operators already implemented on the program and are interacting with other groups in the program (including USC, MIT, and Stanford) to determine how these might perform in a full-scale IU system. A second issue, one that is now becoming the mainstream of the program, is a detailed investigation of the applicability of very large scale integration (VLSI, >50,000 gates/chip) to higher-level operators. Our aims and progress in this area are described below.

1. Introduction

Our work this period on the IU program has been divided into two distinct topics: the performance investigation of the LSI circuits developed for the low-level operators, and an investigation and analysis of the potential of VLSI for intermediate and higher-level operators. We are completing a detailed performance review of the operators developed to-date on the program to characterize the important systems issues, including throughput, accuracy, and dynamic range. This is partially in response to the interest that has developed within the community in applying these circuits. This will also enable us to determine how they might interface with either a host machine for the high level or operate as part of a more comprehensive VLSI system. (To this end, we have had discussions with USC, MIT, and Stanford and have supplied sample circuits.)

The second topic which has now become the central theme of our work centers on the application of VLSI to implement the low-level, intermediate, and high-level processing. Before an effective VLSI architecture can be configured, it is necessary to obtain a complete performance specification of candidate processors together with an analysis of existing algorithms in terms of throughput, data bandwidth, local storage, etc. To this end, we have chosen a line finder, texture processor, and a segmentation scheme as candidate systems. As a base-line, we have taken the algorithms developed by Nevata,[1] Laws,[2] and Olander/Price[3] for each system. In addition, we are investigating computational techniques, such as "residue arithmetic," that may be well suitable to VLSI and may allow increased throughput over conventional binary arithmetic. Details of our progress on each topic are given below.

2. Performance Evaluation of Test Chip III

As we have reported at previous workshops,[4] we have developed a total of 14 LSI primitives aimed specifically at real-time implementation of the low-level operators. These and their effective throughput are listed in Table I.

Table 1. CCD/MOS Circuits Developed on the IU Program

Test Chip Numbers	Algorithms Implemented	Kernel Size	Operations per Pixel	Effective Operation Rate
I	Edge detection	3 x 3	16	80 KOPS
	High-pass spatial filter	3 x 3	18	90 KOPS
	Laplacian	3 x 3	13	65 KOPS
	12 dB/aperture corrector	3 x 3	18	90 KOPS
II	Sobel	3 x 3	16	32 MOPS
	Mean	3 x 3	9	18 MOPS
	Unsharp masking	3 x 3	13	26 MOPS
	Binarization	3 x 3	10	20 MOPS
	Adaptive stretch	3 x 3	12	24 MOPS
III	Laplacian	3 x 3	13	91 MOPS
	Mask programmable convolution	7 x 7	98	686 MOPS
	Programmable convolution	5 x 5	50	350 MOPS
	'Plus' shaped median	5 x 5	625	$\sim 10^3$ MOPS
	Bipolar convolution	26 x 26	1352	$\sim 10^4$ MOPS

To fully characterize this work and provide input to our VLSI study, we are continuing a performance analysis of each of the circuits developed. Particular emphasis is being given to Test Chip III, which has five functions including a 5x5 programmable convolution, a 7x7 mask programmable convolution currently implemented as an edge detector, a 3x3 Laplacian operator, a 5 element sort for median filtering, and a 26x26 bipolar convolution. Preliminary test results earlier[5] demonstrated that the circuits were functionally correct (e.g., the 5x5 kernel was programmable, the sort circuit could perform the sort, and the 26x26 convolution had the proper impulse response). Results included images that were processed by the chip at pixel rates of ∿ 20 kHz. Our current work has been to extend the testing, both in terms of quantitative performance evaluation and in terms of achievable pixel rate. We are currently investigating the following topics: dynamic range of programmable weights on the convolutions, linearity of the transfer functions over the full range of operating conditions, performance evaluation of the sample and hold, dynamic range of the median operator, and the maximum speed of each device. The results of this work will be made available to the community.

3. Development of Intermediate-Level VLSI Image Understanding System

A major goal for this phase of our program is an investigation of the impact and benefits of VLSI technology for image understanding. The complexity of the image analysis and understanding problem clearly indicates that some form of special machinery will be necessary to achieve both the throughput and the memory requirement for even relatively unsophisticated systems with real-time capability. The advent of VLSI and the high-density microelectronic technologies promises to provide some of the answers to our present problems. However, it can be anticipated that many of the basic assumptions used in present and previous generations of processors will not be valid in the era of VLSI technology. A significant example of this is the hitherto paramount aim to reduce the necessary gate count in the processor so as to reduce the machinery cost and increase the reliability. This constraint may well be invalid in future special-purpose processors where the gate density may exceed 10^5/chip. A good case has been made that the data manipulation and interconnects may be the significant burdens in future machines where gates are essentially free and vias cost highly in terms of silicon area, delay times, and design effort.

Our design approach to this problem is shown in Figure 1. It involves determining the algorithms used on present-generation general-purpose machines, from which a directed graph is structured to illustrate explicitly the throughput, data bandwidth, and local storage requirements. An analysis of this structure can then determine the potential conflicts and bottlenecks. At this stage, specific efforts can be made to incorporate the maximum parallelism and concurrency. With the potentially large gate count available with VLSI, this step has great significance, and it is at this point that the interplay between the algorithm and architecture is the greatest. Typically, by going back and re-structuring the algorithm (often by increasing the concurrency), a more uniform data flow can be achieved. The final result of several iterations will hopefully result in an algorithm tailored directly to VLSI. Our studies then continue with an investigation of the hardware commonality between the candidate systems and an essential partitioning in terms of the chip layout, which is then simulated to determine both the resource requirement and the device characteristics needed.

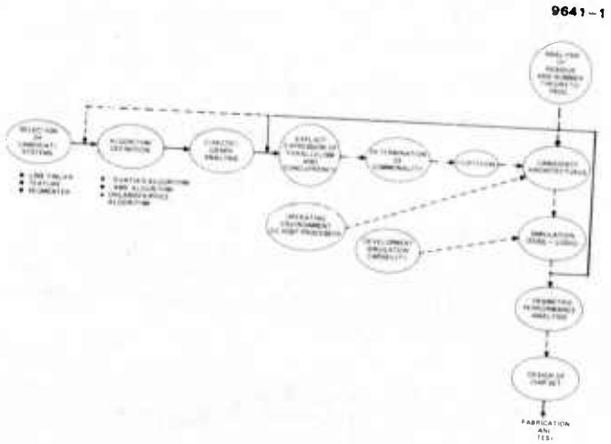


Figure 1. Program Plan for Development of Intermediate Level VLSI Image Understanding System

To support this work we are also developing unconventional architecture techniques and a hardware simulation capability as discussed below.

A. Directed Graph Analysis of Candidate Systems

Three systems involving intermediate level processing were selected for the purpose of developing architectures that utilize commonality between the systems. The systems chosen were a line-finding system, a texture-classification system, and a segmentation system. To be able to perform the directed graph analysis, algorithms must be specified; the following algorithms were chosen:

(1) Nevatia line finder[3]
(2) Laws texture classification system[4]
(3) Ohlander/Price segmenter[5]

The graph analysis was carried out using papers describing the algorithms and talking to the individuals involved in developing them. The graphs for the three systems are shown in

Figures 2, 3, and 4. Constructing the graphs involves identifying logical function blocks and the data inputs and outputs of each block. To indicate the throughput required, the word length and the data rates are indicated on the arcs connecting the function blocks. For example, for the Nevatia line finder in Figure 3, the input data is an eight-bit parallel word and the data rate is NxN pixels per frame multipled by F frames per second.

Once the graphs are constructed, we can begin to see the potential parallelism of the algorithms as well as the common functions between the systems. For example, for the Laws texture system, it is apparent that the processing from the 5x5 convolutions through to the energy measure involves M independent signal paths, indicating that a logical partition would be to put the 5x5 convolution, the normalization, and the large window energy measure on a single chip (if possible). To realize this system, we need only make M copies of the chip. The advantage of this type of partitioning is that the communication to and from the chip is minimized. Several similarities between the systems are made obvious by the graphs. There are parallel convolutions in both the line-finder system and the texture system. A less obvious similarity is between the edge-linker function in the line finder and the connected region finder in the segmentation system. Both functions play the same role of linking data points into a single element, either a line or a region.

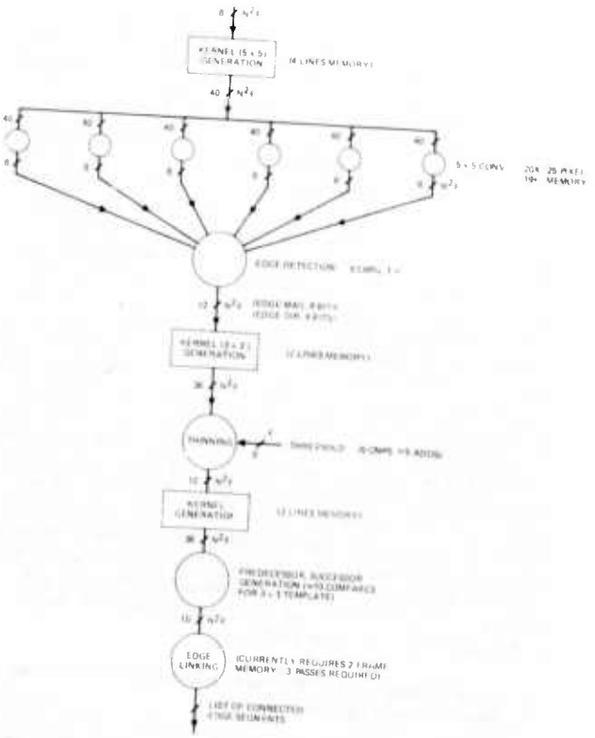


Figure 3. Nevatia Line Finder

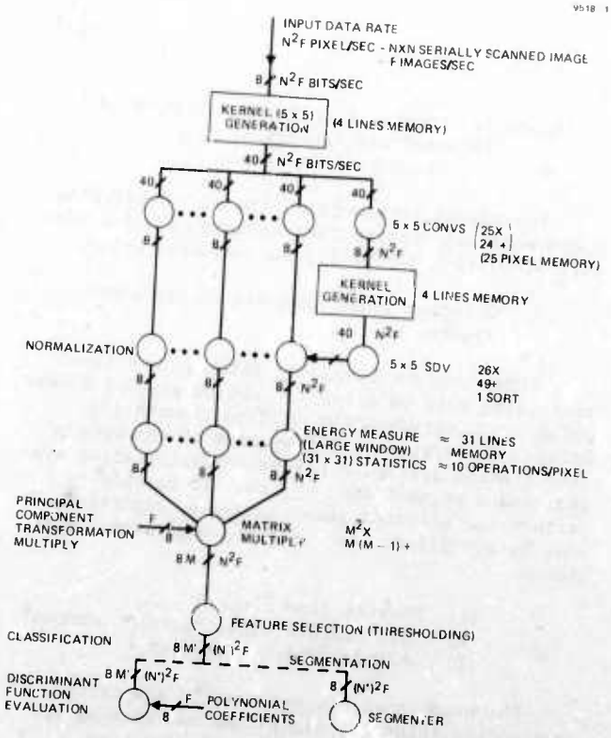


Figure 2. Laws' Texture System

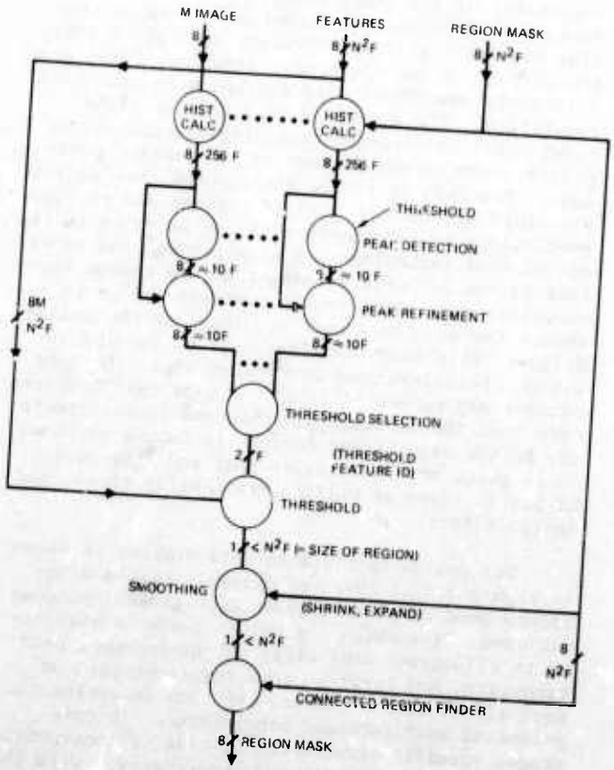


Figure 4. Ohlander Segmenter

B. Residue Techniques for an IU Processor

We are interested in residue arithmetic as a possible candidate for the low-level operations where high throughput is required. Several aspects are intriguing, including the essentially modular nature of the processor, which permits the dynamic range to be changed by simply an additional parallel processing structure using a different base. An attractive feature of this technique is that it eliminates the "carry" concept and the associated propagation delays. Finally, the increased speed available by using a radix higher than the conventional binary structure is advantageous. In this sense, the residue can be considered as a bridge between the high-accuracy, low-speed digital and the much faster but less accurate analog processors.

To perform residue operations, three steps are involved. First, the binary operands must be converted to remainders of several bases (M_i's). Next, these remainders must be added in a modular fashion (i.e., the sum of the two remainders must also be a remainder, which is to say that one must take the remainder of the sum). Finally, the result must be converted back to binary, which seems to be the most difficult of the three steps. In addition to this overhead of conversion, it is difficult to make decisions while in the residue representation since, unlike binary, there is no ordering associated with the representation.

The overhead costs associated with the residue technique for computation makes it inappropriate for many applications. But often it is easier to do several operations with small operands than to do one operation with large operands. So if enough operations can be done between the two conversion steps, the process as a whole could be cheaper to do than the equivalent process using binary arithmetic. For IU work, the bulk of the computation is done at the low level and consists mostly of signal processing requiring no decisions to be made. For algorithms such as a line finder that require several parallel convolutions, there are numerous operations that can be performed in the residue form. Additionally, we are dealing with integer quantities, which are ideally suited to residue techniques. In terms of building hardware, a residue processor would have relatively simple logic performing the bulk of the operations and the complex logic would be concentrated at the input and the output of the system. This is an advantage in terms of reliability and testability.

Our goal for this program is to examine the different systems that we have chosen and to determine the feasibility of using the residue technique. The study will be done in a parametric fashion, so that, by using the parameters of a particular system, one can determine if residue will be an advantage. Areas that we have already begun to look at or will be looking at include conversion logic, dynamic range requirements (rounding in residue makes little sense), logic for adding and multiplying in residue, methods for scaling intermediate results, and methods of representing the numbers in residue to facilitate the logic for calculations.

C. ECSS Simulation

As the density of micro-electronic circuits increases, the systems that can be built on these chips may become very sophisticated. To optimize the design and to be able to predict performance, it is desirable to simulate the operation before the circuits are actually constructed in hardware. For our work on systems design at HRL, we have chosen to use a program developed specifically for computer system simulation at RAND Corporation called ECSS (Extendable Computer Simulation System). ECSS is written in a natural English-like language, and it is both descriptive and flexible to use. It stresses a procedural format and the syntax of the language is constructed so it can compactly define and specify the components of a computer system (storage devices, I/O devices, CPUs, printers) as well as interactions among the components (job scheduling, resource allocation, interprocess communication). To indicate performance and bottlenecks of the computer system being modeled, ECSS outputs statistics showing the jobs activated, percent utilization of the devices, and length of queues for the various resources. The program provides a variety of techniques for modeling systems, and the user, depending on his interests, may focus on modeling program behavior, system I/O transfers, or arbitrary activities running on a particular device. It is built on SIMSCRIPT II, a program language designed for discrete event simulation. Therefore, ECSS has all the provisions of data types and procedural statements of SIMSCRIPT. This allows the user to extend definitions of ECSS structures (devices, jobs, transmissions) and add new commands as desired. Also, when the models provided by ECSS do not meet the requirements of the user, it is possible to modify the source code for ECSS. The service routines are written in modular fashion in SIMSCRIPT, thus facilitating modification.

D. Preliminary Definition of Line-Finder System

Our initial work has been concerned with determining approximately how many gates the line-finder system would require.

The gate count has been divided into gates for memory and gates for random logic. These figures, along with a state-of-the-art number for gates per chip for both random logic and memory, were then used to determine an approximate total number of chips for the system. Additionally, we calculated a chip count for the predicted 1985-1990 technology. This was done for each block on the directed graph. These functions include:

(1) Kernel generation
(2) 5x5 convolutions
(3) Edge detection
(4) Edge thinning
(5) Predesessor and successor generation
(6) P and S memory
(7) Edge linking.

Rough designs were performed for the convolutions, edge detection, and edge thinning to estimate the number of gates. The P and S generation can be estimated by replicating the edge-thinning logic six to ten times, since it is making similar comparisons as for edge thinning but more of them. The logic for edge linking is involved, since it must be able to generate addresses for random access of the P and S memory. In addition, logic for each pass must be generated, since each pass performs a different function. So for the purposes of this sizing, the edge linking was not estimated. Table 1 shows the gate counts for each of the functions. Notice that, for the convolutions, we sized two types, one with no multiplies and one with six multiplies. This is because the algorithm we are sizing has six convolutions, two of which can be scaled such that all the weights are either zero or one, and four which can be scaled such that all but six of the weights are either zero or one.

Table 2. Cate Count for Line-Finder

Function	Cate Count	Number Used	Total
5 Line Kernel	4x512[1]x8 = 16K bits	1	16K bits Memory
5x5 Conv No Multiplies	1.25K	2	2.5K
5x5 Conv 6 Multiplies[2]	13.25K	4	53K
Edge Detection	.35K	1	.35K
3 Line Kernel	2x512x12 = 6K bits	2	12K bits Memory
Edge Thinning	.35K	1	.35K
P and S Generation	3K	1	3K
P and S Memory	512x512x10 = 2.5 Mbit	1	2.5 Mbit Memory

1 - Assumed image size = 512x512
2 - Multiplies accomplished using 8x256 bit ROM.

The current state of the art for MOS technology allows us to put 64K bits of memory or 20,000 gates of random logic on a single ship. Using these figures, we are able to partition the algorithm onto a set of chips and thus calculate a total number of chips for the system. One possible partitioning is as follows:

Chip Functions

1 5 line kernel, 2 convolutions (no multiply)
2-5 One 5x5 convolution (6 multiplies) per chip
6 Edge detection, 3 line kernel, thinning logic
7 3 line kernel, P and S generation
8-47 P and S memory (40 64K chips)
48 Edge linking logic.

The two pacing items (not counting the edge linking logic, which we have not looked at) are the P and S memory and the convolution calculations. The convolution complexity can be simplified by reducing the accuracy required on the weights of the kernel. This would imply that a logic multiplier could be fast enough, and the memory for look up table multiplies would not be needed. The requirement for the P and S memory is embedded in the algorithm, and thus the algorithm would need to be altered if we wanted to reduce this memory.

We can perform this same analysis assuming some future technology. We expect that in several years we will be able to achieve 1 Mbit of memory or 0.5 million gates of random logic on a single chip. This assumes a 1-μm feature size as compared to 5-μm features for current technology. For this case, we get the following partitioning:

Chip Function

1 All random logic and kernel generation memory
2-4 P and S memory.

Our present configurations indicate that we will not be pin limited for this particular operator and hence this partitioning is practical.

It should be emphasized that these estimates are very preliminary and that the effort on these algorithms and the other candidate systems will continue during the next period.

REFERENCES

1. R. Nevatia, K. Ramesh Babu, 'An Edge Detection, Linking and Line Finding Routine', USC I.P.I. Semiannual Report, September, 1978

2. K.I. Laws, 'Textured Image Segmentation', PhD Thesis, USC, January, 1980

3. R. Ohlander, K. Price, D. Raj Reddy, 'Picture Segmentation Using a Recursive Region Splitting Method', Computer Graphics and Image Processing, 1978

4. C.R. Nudd, P.A. Nygaard, S.D. Fouse, T.A. Nussmeier, 'Implementation of Advanced Real-Time Image Understanding Algorithms', Proceedings Image Understanding Workshop, DARPA, April, 1979

5. G.R. Nudd, S.D. Fouse, T.A. Nussmeier, P.A. Nygaard, 'Development of Custom-Designed Integrated Circuits For Image Understanding', Proceedings of Image Understanding Workshop, DARPA, November, 1979.

A "NON-CORRELATION" APPROACH TO IMAGE-BASED VELOCITY DETERMINATION

O. Firschein

Lockheed Palo Alto Research Laboratory
Palo Alto, CA 94304

M. Oron*

Department of Aeronautical Engineering
Technion, Israel Institute of Technology
Haifa, Israel

ABSTRACT

Determination of the ground velocity of a vehicle using time-sequential images is part of the Passive Navigation study dealing with autonomous navigation of a vehicle using passively sensed images. This paper discusses a "non-correlation" intensity gradient approach to velocity determination, and shows in an appendix that such approaches are indeed related to correlation. A low cost "velocity meter" is described that is based on intensity gradients and uses a linear solid state sensor.

INTRODUCTION

The determination of the ground velocity of a slow-flying aircraft using passively sensed images has been one subsystem under investigation in the Passive Navigation Study[1]. Phase correlation or area correlation is often used to determine the displacement between a time sequenced pair of images, and then, knowing the time increment, the relative altitude, and the orientation of the vehicle, one can determine the ground velocity. Prof. M. Oron, Consultant to Lockheed's Independent Research program, suggested a technique for velocity determination that uses intensity gradients rather than correlation. The basic idea is that the displacement can be found using the intensity/time derivative between corresponding pixels in a time-sequenced pair of images (the interframe differences), and the intensity/space derivative for each pixel in a frame (the intra-frame difference). A derivation of this will be given in the description of the velocity meter.

It was later realized that the Oron method is related to other "non-correlation" approaches (2,3,4), particularly those used in TV data compression. Such techniques are applicable when small pixel displacements between frames exist, as in the TV application, or in the case of a slow-flying aircraft sensing the ground scene at TV rates. An analysis of the basic approach due to Limb and Murphy[2] by Dr. W. G. Eppler of LPARL showed that there is indeed a relationship to correlation; his proof is given in Appendix A.

The balance of the paper is devoted to a summary of the velocity meter based on the Oron approach. The full description is given in Ref. 5.

THE GROUND VELOCITY PROBLEM

Continuous on-board determination of ground velocity followed by time integration, can be used for autonomous vehicle navigation. While realizing such a dead-reckoning system, it is necessary to minimize error accumulation, mainly due to changes in the vehicle's attitude. If a two-dimensional imaging device such as a TV camera were used for the velocity determination, it would be necessary to mount it on a gimballed inertial platform similar to those used in accelerometer-based navigation systems. This could result in an instrumentation package which might be more massive, expensive and limited in application than conventional Inertial Navigation Systems (INS), but less accurate. Image-based velocity determination for navigational purposes can, therefore, become practical only if it is realized at much lower cost and weight than INS. This is particularly true when considering the most likely types of aircraft in which such systems would be mounted in order to provide them with autonomous passive navigation[1] capabilities: the miniature Remotely Piloted Vehicles (mRPV), some of which are less expensive than an advanced INS.

In the system described below, simplicity, use of dimensionally small and inexpensive components, and exploitation of other instruments usually installed in an mRPV are emphasized. A solid state electro-optical line sensor, such as the linear array CCD which is inherently small and relatively inexpensive, is used for imaging. Since this sensor is amenable to electronic compensation of attitude changes of autopilot controlled aircraft, as has been recently demonstrated by Oron and Abraham[6], the necessity for electromechanical and gyroscopic stabilization of the optical axis is eliminated, thus avoiding a heavy cost and weight penalty.

An additional advantage of one-dimensional imaging is in the much lower rate of data generation as compared with the two-dimensional case.

---

* During 1979-80 Academic Year: Visiting Scholar at Department of Aeronautics and Astronautics, Stanford University, CA 94305.

This makes it possible to use modest computational power and limited storage memory in the digital signal processing phase. Furthermore, the computational method is based on brightness differences, rather than on two-dimensional correlation, and is not only less scene-dependent, and therefore less limited in scope, but also requires significantly less time and memory.

BASIC CONCEPT

A vertical cross section through an airborne imaging system based on a line sensor containing M elements, each giving rise to a pixel (picture element) of size δ and intensity $I_{n,i}$, is shown in Figure 1. The i-index (i = 1,...,M) designates the position of the element (or pixel) along the sensor axis, x, which lies in the focal plane of the lens, at an angle γ to the longitudinal axis of the aircraft, x_a (see Figure 2). The n-index (n - 1,...,N) designates the pixel's time of occurrence:

$$t_n = n\tau \tag{1}$$

where τ is the exposure time of the sensor between readings. Thus, every τ millisecond the sensor generates M pixels or readings of intensity $I_{n,i}$ which are discrete samplings of a two-dimensional image brightness function, f(x,y), taken along the x-axis at time t_n. Since x and y vary with time because of the aircraft's motion relative to ground (the ground velocity, V), the intensity readings will also vary with time.

It is easier to visualize this variation as being caused by a motion of the whole brightness function f(x,y) in the focal plane in a direction opposite to that of the ground velocity, V. This "focal velocity" is given by:

$$v = -\frac{F}{H} \cdot V \tag{2}$$

where F is the focal length of the lens and H is the aircraft altitude relative to ground. Since F is constant and known and H is measured independently by other on-board instrumentation, the extraction of v from the $I_{n,i}$ readings will make it possible to continuously determine and integrate V with respect to time.

As mentioned above, f(x,y) is an implicit function of t. For those parts of this function which are also continuous and analytic, it is possible to calculate the total time derivative:

$$\frac{df}{dt} = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{df}{\partial y} \cdot \frac{dy}{dt} \tag{3}$$

From Figure 2 and equation (2) the following relations are derived:

$$\frac{dx}{dt} = v_x = -\frac{F}{H} V_x; \quad \frac{dy}{dt} = v_y = -\frac{F}{H} V_y \tag{4}$$

$$v = \sqrt{v_x^2 + v_y^2} \tag{5}$$

$$\tan \rho = \frac{v_y}{v_x} \tag{6}$$

Substituting (4) in (3), an expression for v_x is obtained:

$$v_x = \frac{\frac{df}{dt} - \frac{\partial f}{\partial y} \cdot v_y}{\frac{\partial f}{\partial x}} \tag{7}$$

For the simple case of $v_y = 0$, i.e., when the sensor is aligned exactly along the ground velocity vector ($\rho = 0$), equation (7) becomes:

$$v = v_x = \frac{\frac{df}{dt}}{\frac{\partial f}{\partial x}} \tag{8}$$

The time and space derivatives in (8) can be approximated by brightness difference expressions, designated $\Delta^n I_i$ and $\Delta^i I_n$ respectively. The $\Delta^n I_i$ expression is related to the total time derivative at $x = x_i$ and is calculated from successive intensity readings $(I_{n-1,i}, I_{n,i}, I_{n+1,i} \ldots)$ of the i^{th} element:

$$\frac{df}{dt} = \frac{\Delta^n I_i}{\tau} \simeq \frac{I_{n,i} - I_{n-1,i}}{\tau} \tag{9}$$

Similarly, $\Delta^i I_n$ is related to the partial space derivative, or brightness gradient, calculated at time $t = t_n$:

$$\frac{\partial f}{\partial x} = \frac{\Delta^i I_n}{\delta} \simeq \frac{I_{n,i} - I_{n,i-1}}{\delta} \tag{10}$$

Substituion of (9) and (10) into (8) while accounting for the optical sign reversal between V and v leads to:

$$v_{n,i} = \frac{\Delta^n I_i}{\Delta^i I_n} \cdot \frac{\delta}{\tau} \tag{11}$$

where $v_{n,i}$ is the focal velocity calculated at the i^{th} element at time t_n. In order to simplify (11) and eliminate the minus sign, a negative unit vector velocity, v_p, equal to one pixel, δ, per one exposure time, τ, in the minus x direction is defined:

$$v_p \triangleq -\frac{\delta}{\tau} \tag{12}$$

and substituted into (11) yielding:

$$v_{n,i} = \frac{\Delta^n I_i}{\Delta^i I_n} \cdot v_p \tag{13}$$

A further simplification is obtained by defining a dimensionless focal length velocity coefficient,

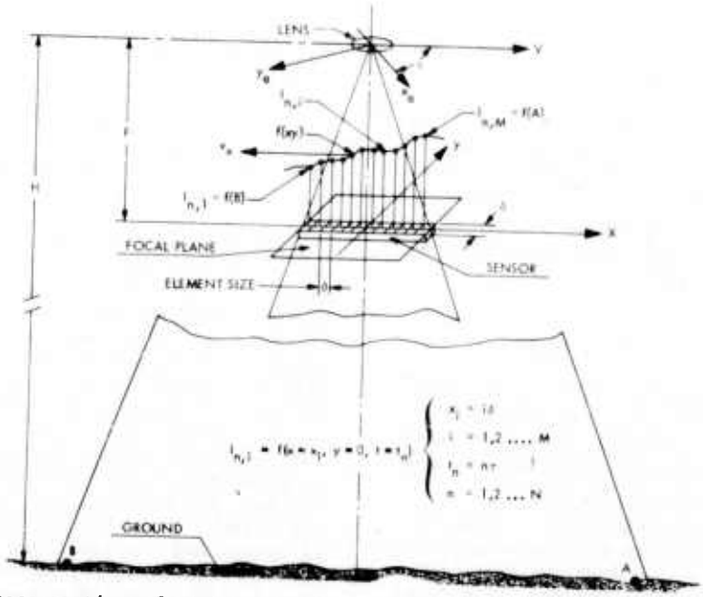


Figure 1. Vertical cross section of airborne imaging system (sensor is parallel to ground velocity vector: $\rho = 0$)

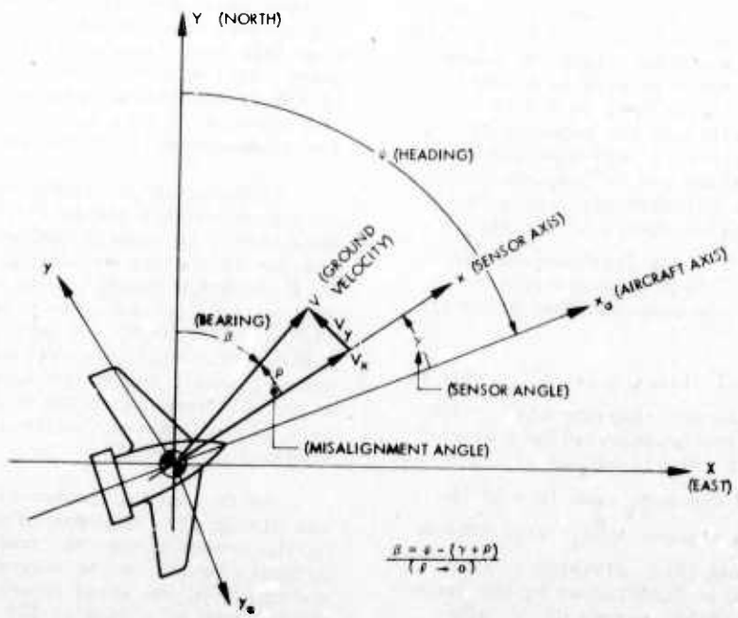


Figure 2. The various axis systems projected onto the ground plane.

$v_{n,i}$ as follows:

$$v_{n,i} \triangleq \frac{v_{n,i}}{v_p} = \frac{\Delta^n I_i}{\Delta^i I_n} \qquad (14)$$

Using equation (14), a total of $(M - 1)\cdot(N - 1)$ values of $v_{n,i}$ are obtained during one period, T, of velocity estimation, where T is given by:

$$T = N \cdot \tau \qquad (15)$$

If M and N are large enough, a statistically satisfactory distribution of the $v_{n,i}$ values is achieved, enabling a good estimate of the true value of v. One simple such estimate would be the arithmetic mean:

$$\bar{v} = \frac{1}{(M-1)(N-1)} \sum_{i=2}^{M} \sum_{n=2}^{N} \frac{\Delta^n I_i}{\Delta^1 I_n} \qquad (16)$$

Thus, at each pixel in a frame, the ratio of the time derivative to the space derivative is determined, and the ratios are averaged over the frame. This differs from the Limb and Murphy approach[2] in which the time derivative and the space derivative are averaged separately, and then the ratio is obtained by dividing these two averages.

PROPOSED INSTRUMENTATION SETUP

For the implementation of the basic concept described above, an instrumentation setup which includes some components found in RPVs is proposed. Employing the CCD line sensor (1024 elements) used by Oron and Abraham[6] in their system (δ = 12.5 μm, τ = 2 msec) with optics similar to theirs (F = 100 mm) but arranged in an imaging geometry shown in Figure 1; a negative unit-pixel velocity v_p = -6.25mm/sec is obtained. The focal velocity for typical flight conditions (V = 80 kt, H = 3000 ft) is v = -4 mm/sec and the velocity coefficient is ν = 0.64. As the ground velocity will vary between 40 knots to 120 knots, the variation of this coefficient will be ν = 0.32 - 0.96. In other words, the translation along the sensor between two adjacent exposures will be between about a third of a pixel and just less than a whole pixel. The translation per exposure due to pitch and roll instability is less than 10% of that, but its accumulation can be compensated for using signals from the on-board vertical gyro installed in the autopilot system of the RPV as previously described[6]. For that purpose only 512 readings out of the 1024 elements will be actually processed in the computational procedure for determining ν.

The 511 interpixel differences, $\Delta^1 I_n$, will be computed on-line during one exposure time τ (or the order of 2 msec) and transferred to a special register together with the 512 values of I_i. At the end of the second exposure time (n = 2) the 511 interexposure differences $\Delta^n I_i$ will also be computed using equation (9). Division of the latter set of brightness differences by the former will yield a maximum of 511 values of ν after every exposure. In reality, the computation unit will also include a logical circuit which will discard many of the above computations: the $\Delta^n I_i$ denominator values which are lower than a certain threshold, will first be eliminated in order to prevent erroneously high ν results caused by scenery which is too low in contrast ("flat") to yield a meaningful spacial brightness gradient. Secondly, very high or abrupt gradients ("sharp edges") will also be discarded since they represent discontinuities in the f(x,y) brightness function or in its first derivative and, therefore, belong to non-analytical portions of f(x,y). If a period of about T = 80 msec (N = 40) is chosen, over 20,000 values of ν will be computed, and at least 10,000 of them are expected to be valid for each period. This is a relatively large and statistically controlled population of results which assures a Gaussian distribution, provided that only random errors (or non-correlated noise) occur in the system while all the systematic errors are eliminated.

An important systematic or non-random error may arise as a result of misalignment between the sensor and the direction of the ground velocity vector as can be seen from comparison of equations (7) and (8). To prevent such misalignment, the sensor could be mounted on a special holder which will be rotated in the focal plane around an axis of rotation which is perpendicular to the earth surface or parallel to the z axis of the aircraft. The rotational motion can be provided by a small stepping motor which need not be faster than 100 Hz (10 msec per step) or more accurately positioned than within 1°. Thus, if the direction of motion is known, and the time spent at each angle is 80 + 10 = 90 msec, the system will during one second check about 10° of angle, 5° to each side of this direction. After such angular scanning, about 40 different sets of ν computations, each consisting of more than 10,000 valid values will be obtained. The set which will have the best statistical distribution of its values around the mean (the most Gaussian-like and symmetric histogram), will be the one with the smallest systematical or directional bias error and, therefore, the sensor will then be most closely aligned with the ground velocity direction.

Mounting of the electro-optical sensor on a rotational holder has an additional practical advantage: it enables compensation for sudden changes in the yaw or heading angle ψ measured by an on-board compass. The output of this instrument will be processed to provide a correctional signal which can be fed into the stepping motor controller. Thus, whenever a sudden change in yaw occurs, mainly due to gusts or wind the stepping motor will compensate for it immediately (a 100 Hz rate is much faster than required by the flight dynamics of an mRPV).

So far only direction-following or tracking was discussed. The problem of initially setting up the sensor along the direction of the ground velocity vector can be solved within 5 to 10 seconds. In the worst possible case, a "brute force" scan at a rate of 20° of angle per second will require 9 seconds to check all possible directions (180° of angle) to an accuracy of 2°, but it is more likely that the ground velocity direction will be found sooner than that. In fact, a strategy of starting from the heading direction of the aircraft and scanning systematically on its both sides, will lead much faster to the plane's bearing angle β (which gives the ground velocity direction relative to the North) since in most cases the heading and bearing of a mRPV are not too widely separated from each other.

DISCUSSION

A computer-simulated experimental investigation was carried out to test these ideas. The simulation, described in Ref. 5, indicates that while working at υ values close to unity, it is possible to achieve high directional sensitivity as well as magnitude accuracy. In order to approach such υ values it is necessary to introduce a change in τ as V varies; thus after V has been calculated, a signal is fed-back to the sensor controller which changes its frequency between 200 KHz and 500 Hz.

The overall accuracy of the ground velocity determination, and hence of the whole dead-reckoning navigational system, is perhaps more critically dependent on the auxiliary readings of the altitude, H, and the heading angle, ψ, than on the new electro-optical sensor-based part of the system. If a passive absolute altitude barometric altimeter is used, an accuracy of approximately 1% can be achieved, however, the uncertainty in terrain elevation data prestored in the system and subtracted from H to yield relative altitude, may raise that figure to a 2% level which for the $\upsilon \simeq 0.9$ range is worse than the v accuracy (about 1%). Experiments show that it is possible to deduce that an accuracy of about 1^{o} can be achieved for the angle of velocity, compatible with the compass ψ readings. Further experiments with hardware rather than computer simulated motion are needed to establish whether a better accuracy in γ can be accomplished justifying the use of more accurate instrumentation for ψ measurements.

In summary, it can be stated that ground velocity determination on-board mRPVs using electro-optical line sensors in combination with existing instruments is not only feasible, but quite practical, sufficiently accurate and not too expensive (in terms of weight and cost). The velocity values are computed at a rate of between 10 to 5 per second, which is much higher than necessary for the relatively slow flight dynamics of a mRPV. This enables us, by using fairly simple filtering and prediction techniques in the navigator prior to integration, to raise the level of confidence in this dead-reckoning navigation system as well as to overcome short "dark" periods when, due to cloud coverage, or extremely "flat" scenery, no velocity computations can be made. In this context it should be emphasized that the method proposed here is by no means limited to the visual spectrum; any electro-optical line sensor can be used, thus perhaps expanding the range of applicability to include overcast days as well as night operation.

ACKNOWLEDGMENTS

The experimental work was performed in the Signal Processing Laboratory of the Lockheed Palo Alto Research Laboratory. We would like to thank Dr. J. J. Pearson, head of the group, Dr. W. G. Eppler, Dr. K. Dutta, and Mr. C. D. Kuglin for many helpful discussions and suggestions. One of us (M.O.) would like to thank Professor D. Debra, head of the Flight Control and Instrumentation Laboratory of the Department of Aeronautics and Astronautics, Stanford University for his kind hospitality and most constructive comments on this presentation. The authors would like to express their gratitude to L. Staley for speed and accuracy in the preparation of the several drafts and the final manuscript.

REFERENCES

1. O. Firschein, D. Gennery, D. Milgram and J. J. Pearson, "Progress in Navigation Using Passively Sensed Images," Image Understanding Workshop, Menlo Park, CA, April 1979.

2. J. O. Limb and J. A. Murphy, Estimating the Velocity of Moving Images in Television Signals, Computer Graphics and Image Processing, 1975, 311-327.

3. C. Cafforio and F. Rocca, Methods for Measuring Small Displacements of Television Images, IEEE Trans. Inform. Theory, IT-22, No. 5 (Sept. 1976) 573-579.

4. A. N. Netravali and J. D. Robbins, Motion-compensated Television Coding, The Bell System Technical Journal, Vol. 58, No. 3, March 1979, 631-670.

5. M. Oron and O. Firschein, Airborne Ground Velocity Determination by Digital Processing of Electro-Optical Line Sensor Signals, Proc. SPIE, Vol. 219, 1980 (SPIE Technical Symposium, Los Angeles, CA, Feb. 1980).

6. Oron, M., and Abraham, M., "Analysis Design and Simulation of Line Scan Aerial Surveillance Systems," Proc. SPIE, Vol. 219, 1980. (SPIE Technical Symposium, Los Angeles, CA, Feb. 1980).

APPENDIX A

RELATION OF CORRELATION TO THE FRAME-DIFFERENCE, ELEMENT-DIFFERENCE METHOD OF VELOCITY DETERMINATION IN TELEVISION SIGNALS

W. G. Eppler

Lockheed Palo Alto Research Laboratory, Palo Alto, CA 94304

A1. INTRODUCTION

Limb and Murphy[2] describe a method for estimating the velocity of moving images in television signals. The displacement between two frames, for small displacements, is given by the equation,

$$\hat{x} = \frac{\Sigma|FDS|}{\Sigma|EDS|} \qquad (A\text{-}1)$$

where $\hat{x}$ is the displacement between the images

FDS is the frame difference (the intensity difference between two successive frames at a particular pixel)

EDS is the element difference (the difference between an element and its left neighbor)

Later papers by other authors[3,4] provide extensions to the original Limb/Murphy equations. This appendix shows the relation of correlation to the Limb/Murphy results.

A2. DERIVATION OF THE CORRESPONDENCE

First we assume that the correlation curve is linear between 0 and 1 sample interval, as shown in Fig. A-1.

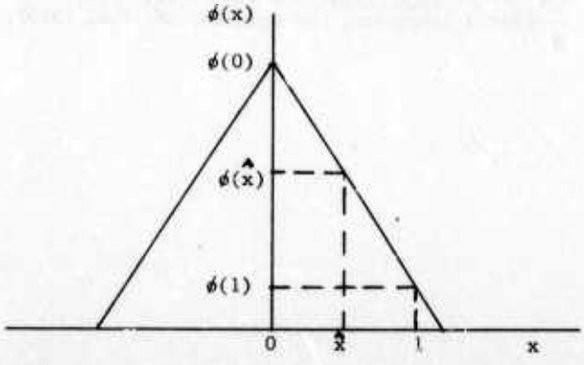


Fig. A-1 Assumed Linear Correlation Curve for $0 \le x \le 1$

If we know the correlation values at $\phi(0)$, $\phi(x)$, and $\phi(1)$, we can interpolate the $\hat{x}$ value as

$$x = \frac{\phi(0) - \phi(\hat{x})}{\phi(0) - \phi(1)} \qquad (A\text{-}2)$$

Now we substitute expressions for the three correlation values, noting that $\phi(0)$ can be written in two ways, since the image is stationary:

$$\phi(0) = \sum_i I^2(i) = \sum_i I^2(i + x) \qquad (A\text{-}3)$$

$$\phi(\hat{x}) = \sum_i I(i)\, I(i + \hat{x}) \qquad (A\text{-}4)$$

$$\phi(0) - \phi(\hat{x}) = \frac{1}{2}\sum_i [I^2(i) - 2I(i)\, I(i + \hat{x}) + I^2(i + \hat{x})]$$

$$= \frac{1}{2}\sum_i [I(i) - I(i + \hat{x})]^2 \qquad (A\text{-}5)$$

$$\hat{x} = \frac{\sum_i [I(i) - I(i + \hat{x})]^2}{\sum_i [I(i) - I(i + 1)]^2} \qquad (A\text{-}6)$$

To the extent that $[\ ]^2 \approx |\ |$

$$\hat{x} \approx \frac{\sum_i |I(i) - I(i + \hat{x})|}{\sum_i |I(i) - I(i + 1)|} \qquad (A\text{-}7)$$

$$\sum_i |I(i) - I(i + 1)| = \sum_i |EDS| \qquad (A\text{-}8)$$

To the extent that the displacement is in the x-direction only:

$$\sum_i |I(i) - I(i + \hat{x})| = \sum_i |FDS| \qquad (A\text{-}9)$$

Substituting A-9 and A-8 in A-7, we obtain,

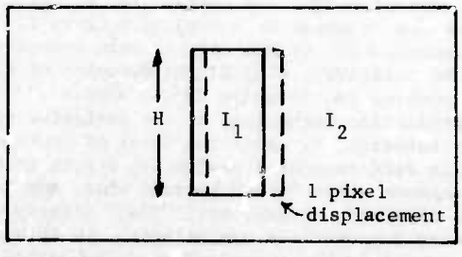


Fig. A-2 Effect of the One Pixel Horizontal Displacement for a Simple Figure

(Limb and Murphy)
$$\hat{x} \approx \frac{\sum_i |FDS|}{\sum_i |EDS|} \qquad (A\text{-}10)$$

Thus, the Limb/Murphy algorithm amounts to subpixel interpolation of the correlation peak in the matchpoint neighborhood; the range of the algorithm is the correlation distance. $|EDS|$ controls the correlation function slope, and depends on the projected lengths and the intensity differences across vertical boundaries; in Fig. A-2 the displaced rectangle of height H results in $|EDS| = 2H|I_1 - I_2|$.

BOOTSTRAP STEREO

Marsha Jo Hannah

Lockheed Palo Alto Research Laboratory
Department 52-53, Building 204
3251 Hanover Street
Palo Alto, CA 94304

ABSTRACT

Over the past two years, Lockheed has been working in navigation of an autonomous aerial vehicle using passively sensed images. One technique which shows promise is bootstrap stereo, in which the vehicle's position is determined from the perceived locations of known ground control points, then two known vehicle camera positions are used to locate corresponding image points on the ground, creating new control points. This paper describes the components of bootstrap stereo - camera calibration, new landmark selection, techniques for efficient control point matching from image to image, and control point positioning.

INTRODUCTION

Before the advent of sophisticated navigation aids such as radio beacons, barnstorming pilots relied primarily on visual navigation. A pilot would look out the window of his airplane, see landmarks below him, and know where he was. He would watch the ground passing beneath him and know how fast and in what direction he was moving. Unless the ground was obscured by clouds or darkness, he did quite well at navigating toward his destination.

Today, there exist applications for which a computer implementation of this simple, visually oriented form of navigation would be useful. One scenario hypothesizes a small, unmanned vehicle which must fly accurately from its launch point to its target under possibly hostile circumstances. This must be accomplished without relying on external signals (which could be jammed) or emitted radiation (which could be used to track and destroy the vehicle). Other options which have been considered and rejected include a simple pre-programmed flight plan (gross errors in course could accumulate from unpredictable wind effects) and a high quality inertial guidance system (which would add excessive weight and expense).

It is felt that the current state of the art in artificial intelligence and image processing, coupled with the availability of small solid state sensors and microprocessors, will enable us to use passively sensed terrain imagery in a visual navigation system small enough to be flown in an autonomous aerial vehicle (AAV). To prove feasibility, we are implementing and refining these techniques in computer software.

Our overall approach to the problem involves providing the vehicle with a Navigation Expert having approximately the sophistication of an early barnstorming pilot. This expert will navigate partly by its simple instruments (altimeter, airspeed indicator, and attitude gyros), but mostly by what it sees of the terrain below it.

The Navigation Expert will consist of an Executive that weighs evidence and makes final decisions, and a group of Specialists (see Figure 1). Each of these Specialists provides information on its area of expertise, along with measures of confidence in its results to aid the Executive in resolving contradictory opinions from two or more Specialists. The Specialists identified to date include an Instruments Specialist to provide images and flight instrument readings, a Dead Reckoning Specialist to estimate position from past navigation data, a Landmarks Specialist to recognize checkpoint landmarks, and a Stereo Specialist to perform a variety of stereo photogrammetric tasks.

This report covers one aspect of the Stereo Specialist, a technique which we call bootstrap stereo.

THE BOOTSTRAP STEREO CONCEPT

Given a set of ground control points with known real-world positions, and given the locations of the projections of these points onto the image plane, it is possible to determine the position and orientation of the camera which collected the image, a process known to traditional photogrammetrists as space resection [T]. Conversely, given the positions and orientations of two cameras and the locations of corresponding point-pairs in the two image planes, the real-world locations of the viewed ground points can be determined, a process known as space intersection [T]. Combining these two techniques iteratively produces the basis for bootstrap stereo.

Figure 2 shows an AAV which has obtained images at three points in its trajectory. The bootstrap stereo process begins with the set of

landmark points, a and b, whose real-world coordinates are known. (In reality, at least four points would be needed; only two are shown here to simplify the diagram). From these, the camera position and orientation is determined for the image frame taken at Time 0. Standard image-matching correlation techniques [H] are then used to locate these same points in the second, overlapping frame taken at Time 1. This permits the second camera position and orientation to be determined.

Because the aircraft will soon be out of sight of the known landmarks, new landmark points must be established whenever possible. For this purpose, "interesting points" -- points with a high likelihood of being matched [M] -- are selected in the first image and matched in the second image. Successfully matched points have their real-world locations calculated from the camera position and orientation data, then join the landmarks list. In Figure 2, landmarks c and d are located in this manner at Time 1; these new points are later used to position the aircraft at Time 2. Similarly, at Time 2, new landmarks e and f join the list; old landmarks a and b, which are no longer in the field of view, are dropped from the landmarks list.

Once initialized from a set of known landmarks, bootstrap stereo has four components, which we will discuss in the following sections:

1) Camera Calibration -- determining the camera position and orientation from known ground control points.

2) New Landmark Selection -- choosing potential new ground control points.

3) Point Matching -- pairing a point in one image with its corresponding point in a second, overlapping image.

4) Control Point Positioning -- locating points on the ground, given their positions in two images and the relevant camera positions and orientations.

CAMERA CALIBRATION

Given a set of ground control points with known real-world positions (Xi,Yi,Zi), and given the locations of the perceived locations of these points on the image plane (Ui,Vi), it is possible to determine the position (XO,YO,ZO) and orientation (HEADING,PITCH,ROLL) of the camera which took the imagery [D&H], [T]. This is accomplished by a least-squares solution of a set of collinearity condition equations, effectively minimizing the mean of the errors between each image plane point (Ui,Vi) and the projection (Ui',Vi') of that point's real-world location (Xi,Yi,Zi) onto the image (see Figure 3). Because the equations are highly nonlinear, a solution is usually sought by iterating on a linearization of the problem [G]. The solution is initialized from camera orientation data provided by the Instruments Specialist and a position estimate from the Dead Reckoning Specialist.

This technique is somewhat sensitive to invalid points which may appear in its data set. Consequently, each camera solution should be checked to see if any of the points are contributing excessively to the residual error. Such points should be removed from the data set and the solution redone, to avoid major errors. Indeed, this editing process usually has to be iterated to obtain maximum data reliability.

A promising technique under development [F&B] forms analytically exact camera position and orientation models from subsets of the point data, then evaluates the points and potential models together, before refining the least-squares model from the reliable points, as above. This technique appears to be an improvement, both computationally and in terms of accuracy, to the present method, and may well be the method of choice for the eventual bootstrap stereo package. Since this method is still under development, we are proceeding with the more standard technique in our feasibility study.

NEW LANDMARK SELECTION

Because the aircraft rapidly moves beyond the known landmarks, new landmark points must constantly be established. For this purpose, "interesting points" -- points with a high likelihood of being matched [M] -- are selected in the old image of each pair, then matched with their corresponding points in the new image and located on the terrain.

Matching is done on the basis of the normalized cross correlation between small windows of data (typically 11 × 11) around the two points in question. If the window to be matched contains little information, it can correlate reasonably well with any other area of similar low information. To avoid mismatches from attempting to use such areas, the simple statistical variance of the image intensities over the window

$$\text{var} = \underset{ij}{\text{MEAN}}(\text{INT}(i,j) - \text{MEAN}(\text{INT}))^2$$

was used as an early measure of information [H], with only areas of high information being acceptable candidates for matching.

Matching also has trouble with strong linear edges, since an otherwise featureless area containing a strong edge will match equally well anywhere along the edge. To reject such areas, the notion of directed variance was introduced [M]. Four quantities are calculated over the window:

$$\text{dirvar1} = \text{MEAN}(\text{INT}(i,j) - \text{INT}(i+1,j))^2$$

$$\text{dirvar2} = \text{MEAN}(\text{INT}(i,j) - \text{INT}(i,j+1))^2$$

$$\text{dirvar3} = \text{MEAN}(\text{INT}(i,j) - \text{INT}(i+1,j+1))^2$$

$$\text{dirvar4} = \text{MEAN}(\text{INT}(i+1,j) - \text{INT}(i,j+1))^2$$

The directed variance is then defined to be the minimum of these four quantities. Points with poor visual texture will have low directed variance because adjacent samples differ little in any of the directions. Points with linear edges will show low directed variance in the direction of the edge. Conversely, points with high directed variance should avoid these defects. Thus, "interesting points" were defined to be local maxima in this "interest operator", directed variance.

We have developed another indicator of the presence of an edge in the window -- the ratio of the directional variances, taken in perpendicular pairs. This measure takes advantage of the fact that a window with a strong edge will have much greater information content across the edge than along it. Because this measure does not give an indication of low information, we have combined it with ordinary variance to form an interest measure we call edged variance.

$$evar = var * MIN\left(\frac{dirvar2}{dirvar1}, \frac{dirvar1}{dirvar2}, \frac{dirvar4}{dirvar3}, \frac{dirvar3}{dirvar4}\right)$$

Figure 4 shows the application of these three interest measures for a sequence of three images taken over the Night Vision Lab terrain model. The images in the first column have overlays showing the peaks in ordinary variance; the second column shows peaks in directed variance; the third column shows peaks in edged variance. Note that ordinary and directed variance find points along the strong, irregular edges which mark the tree/grass boundaries, and ignore open areas having more subtle features. Edged variance, on the other hand, gives a combination of strong and subtle features, while avoiding excessively plain areas. For the selection of new landmarks, edged variance is the interest measure of choice.

POINT MATCHING

The actual matching of points in an image pair is done by maximizing normalized cross correlation over small windows surrounding the points. Given an approximation to the displacement which describes the match, a simple spiraling grid search is a fairly efficient way to refine the precise match [H]. To provide that initial approximation, we have employed a form of reduction matching [H], [M].

As shown in Figure 5a, we first create a hierarchy of N-ary reduction images. Each $N \times N$ square of pixels in an image is averaged to form a single pixel at the next level. (For the example shown, N = 3). This reduction process is repeated at each level, stopping when the image becomes approximately the size of the correlation windows being used.

Matching then begins at the smallest images. A window centered on the image is matched via the spiral search, beginning at the center of the second image. Thereafter, each matched point spawns four points around itself, offset by half a window radius along the diagonals of the window. These are mapped down to the next level of images, carrying their parent's displacement (suitably magnified) as their suggested match approximation. These points then have their displacements refined by a spiraling search before spawning new points. This process (illustrated in Figure 5b) continues until the largest images are reached, effectively setting up a grid of control points for matching.

Having this initialization, we intended that further matches be approximated from the displacement of the nearest grid control point, then refined via the spiral search. We discovered, however, that an occasional point could be lost in the process of carrying the matches down the image hierarchy, either because its match disappeared over the edge of the image, because of low information in an area, or because relief-induced distortion caused a match to be unreliable (as determined by autocorrelation thresholding [H]). Consequently, using the closest point required searching through the grid control points to determine the closest valid one.

To avoid this, we chose to approximate the displacement in a different way. Aerial imagery usually does not present any reversals in displacement, so it is reasonable to approximate the DX and DY components of the displacements by fitting polynomials in X and Y to them [Q]. For each further point to be matched, its position in the first image is used to evaluate the two polynomials, producing an estimate of the position of the matching second image point.

When we used first-order polynomials for this approximation, the residual errors were on the order of a pixel, and reliable matches which had been initialized from these polynomials differed by as much as 3 pixels from the predicted displacement. Using second-order polynomials resulted in residual errors on the order of half a pixel, and reliable matches differed by less than 2 pixels from the predicted displacement. Since this was deemed adequate for initializing the local match search, higher-order polynomials were not tried.

In our implementation of bootstrap stereo, reduction matching is used to determine approximate registration of the images and to initialize the second-order match prediction polynomials. Matching of old landmarks and of interesting points to create new landmarks uses these polynomials to predict an approximate match, which is then refined by a local search. Autocorrelation thresholding is used to test the reliability of the match, then points are located more closely than the image grid permits by parabolic interpolation of the X- and Y-slices of the correlation values.

CONTROL POINT POSITIONING

Given the positions and orientations of two cameras and the locations of corresponding point-pairs in the two image planes, the real-world locations of the viewed ground points can easily be determined [M&H], [T]. The vectors from the focal points of the cameras through the respective

image plane points are simply projected into space (see Figure 6). Since these rays rarely intersect exactly, we find their points of closest approach and average them.

If the difference is large or the real-world point is unreasonably different from its neighbors, the point is rejected as having resulted from a bad match. Otherwise, this point joins the list of control points for future matching and camera calibration.

AN EXAMPLE

In Figure 7, we present an example of the control-point handling portion of bootstrap stereo. The original data set, a sequence of 3 images from a Night Vision Laboratory tape, is shown in Figure 7a.

Figure 7b shows the interesting points in the first image, indicated by + overlays. If these were the control points from a landmark processor, we would use them to locate the first camera. These landmark points are then matched with their corresponding points in the second image; Figure 7c shows the successful matches overlaid on the first and second images. From the image plane positions of these points, the position and orientation of the second camera are determined.

Next, the areas of the second image which were covered by matches are blocked out and interesting points are found in the uncovered areas, as seen in Figure 7d. The old landmark points and the interesting points are then matched in the third image, as shown in Figure 7e. The old control points from the second image are used to calibrate the third camera; the camera calibrations are then used to locate the matched interesting points on the ground, forming new control points.

These last two steps are repeated for subsequent pairs of images in longer sequences.

VULNERABILITY

The bootstrapping process is far from infallible. The errors to which it is vulnerable fall into roughly four categories:

1) Loss of overlapped imagery.
2) Errors in matching control points.
3) Errors in camera calibration.
4) Errors in control-point positioning.

The bootstrapping process could lose its needed overlap in imagery if the terrain over which the AAV is flying is obscured by clouds. If the terrain is essentially featureless (for example, a large body of water or a desert), then the bootstrapper will be unable to find and match sufficient control points to continue. Similarly, several dropped frames resulting from a temporary equipment failure could cause the bootstrapping process to abort.

A human navigator faced with clouds or featureless terrain would simply shift mental gears and fly on instruments and dead reckoning until more favorable terrain was found. He would then attempt to locate new landmarks from which to re-orient himself. Mimicking this, when stereo bootstrapping loses its overlapped imagery, the Stereo Specialist just reports failure to the Navigation Expert, which then proceeds to rely on its Dead Reckoning Specialist until its Landmarks Specialist can recognize some new landmarks from which to re-orient the Stereo Specialist for bootstrapping. Until then, the Stereo Specialist processes any available imagery to extract ground velocity for the Dead Reckoning Specialist.

The other three problems rarely cause the bootstrapping process to fail completely. Instead, they interact to create errors in the vehicle positions determined by bootstrapping. Since these are somewhat inevitable, it is anticipated that the Landmarks Specialist will be invoked periodically to search for checkpoint landmarks. Course corrections will then be determined from these checkpoints, and the bootstrapper will be re-initialized.

Gross errors in match, which can occur due to repetitive textures or moving objects, are likely to be caught by the autocorrelation thresholding or by the depth consistency or camera model consistency requirements. Small errors in match, such as would result from improper sub-pixel registration of the data, can slip through; these will bias the camera calibrations and control point locations slightly.

Errors in camera calibrations result either from errors in the data or from insufficient precision in the calibration calculations. The latter can be designed out of the system by insuring that the processor has sufficient word-length and/or floating-point precision to handle matrix inversion. Errors in the data are most likely to affect the camera position, as its orientation is fairly well known from the vehicle orientation reported by the Instrument Specialist. Techniques exist [T] for the adjustment of the image data points along with the camera parameters, to produce more consistent results.

Errors in the positions of the original landmarks will be propagated through the bootstrapping chain. Such errors should be static, however, and should only result in a small perturbation in the vehicle location. Errors in control point positioning are interrelated with errors in the match point location and the camera calibration. Using the redundancy inherent in multiple images [M] can resolve some of these uncertainties.

The manner in which these errors accumulate is complex and not readily amenable to analytic examination. One of our tasks in the coming months will be to examine these errors in simulation. We plan to generate sets of known ground points and known camera locations and orientations. For each camera position, the visible ground points will be mathematically projected into the

simulated focal plane, and the camera localization and control point positioning portions of the bootstrapping process will be run. It will be possible to perturb the data at each step in the process, so we can analyze the effects of various errors on the bootstrap procedure by comparing the calculated camera and checkpoint positions to the true positions which generated the original data.

CONCLUSIONS

When an autonomous aerial vehicle must navigate without using external signals or radiated energy, a visual navigator is an enticing possibility. We have proposed a Navigation Expert capable of emulating the behavior of an early barnstorming pilot in using terrain imagery. One tool such a Navigation Expert could use is bootstrap stereo. This is a technique by which the vehicle's position is determined from the perceived positions of known landmarks, then two known camera positions are used to locate real-world points which serve as new landmarks.

The components of bootstrap stereo are well established in the photogrammetry and image processing literature. We have combined these, with improvements, into a workable system. We are continuing to work on the error analysis, to determine how the errors propagate and accumulate.

REFERENCES

[D&H] Duda, R. O. and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, New York, New York, 1973.

[F&B] Fischler, M. A. and R. C. Bolles, "Random Sampling Consensus", Proceedings: Image Understanding Workshop, College Park, Maryland, April 30, 1980.

[G] Gennery, D. G., "A Stereo Vision System for an Autonomous Vehicle", Proceedings of the 5th IJCAI, Cambridge, Massachusetts, 1977.

[H] Hannah, M. J., Computer Matching of Areas in Stereo Imagery, Ph.D. Thesis, AIM #239, Computer Science Department, Stanford University, California, 1974.

[M] Moravec, H. P., "Visual Mapping by a Robot Rover", Proceedings of the 6th IJCAI, Tokyo, Japan, 1979.

[Q] Quam, L. H., Computer Comparison of Pictures, Ph.D. Thesis, AIM#144, Computer Science Department, Stanford University, California, 1971.

[T] Thompson, M. M., Manual of Photogrammetry, American Society of Photogrammetry, Falls Church, Virginia, 1944.

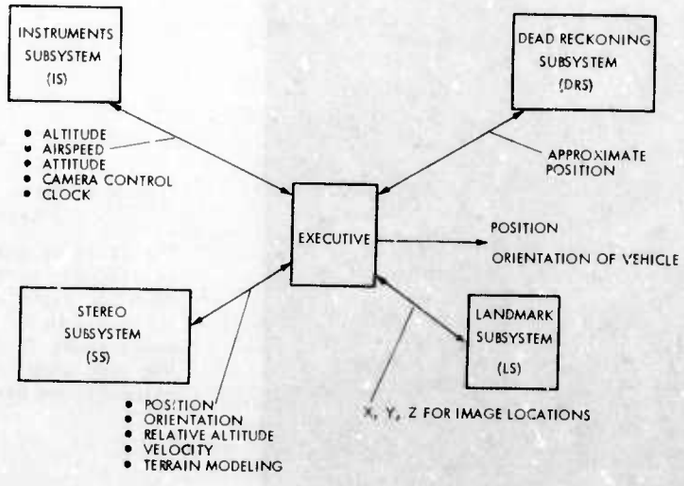
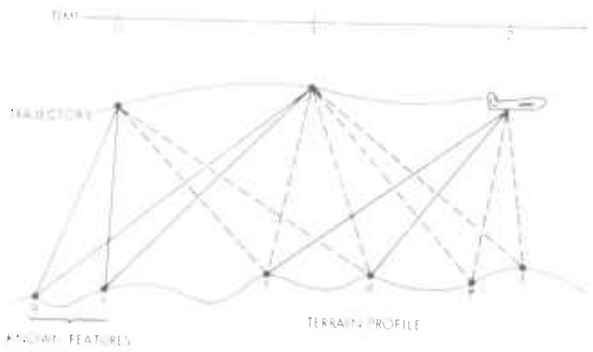


Figure 1 Components of the Navigation Expert.



TIME	POSITION OF CRAFT DERIVED FROM	DETERMINE POSITION OF
0	a, b	—
1	a, b	c, d
2	c, d	e, f

Figure 2 Navigation Using Bootstrap Stereo.

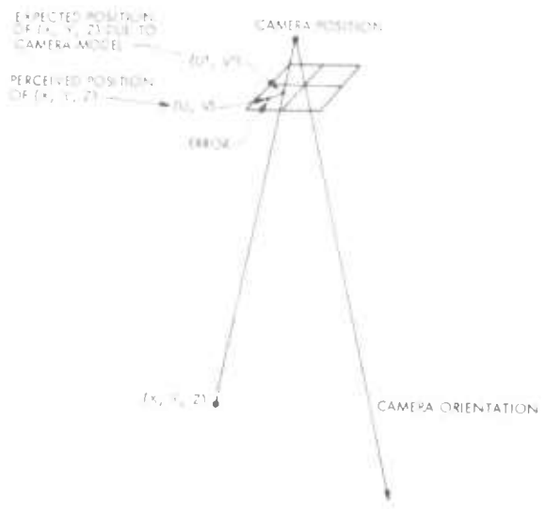


Figure 3 Calibration of Camera Position and Orientation.

The position and orientation of the camera is derived by searching for the camera model parameters which minimize (over a set of points) the error between the perceived position (U,V) of a point (X,Y,Z) and the position to which it would be projected (U',V') if this were the correct model.

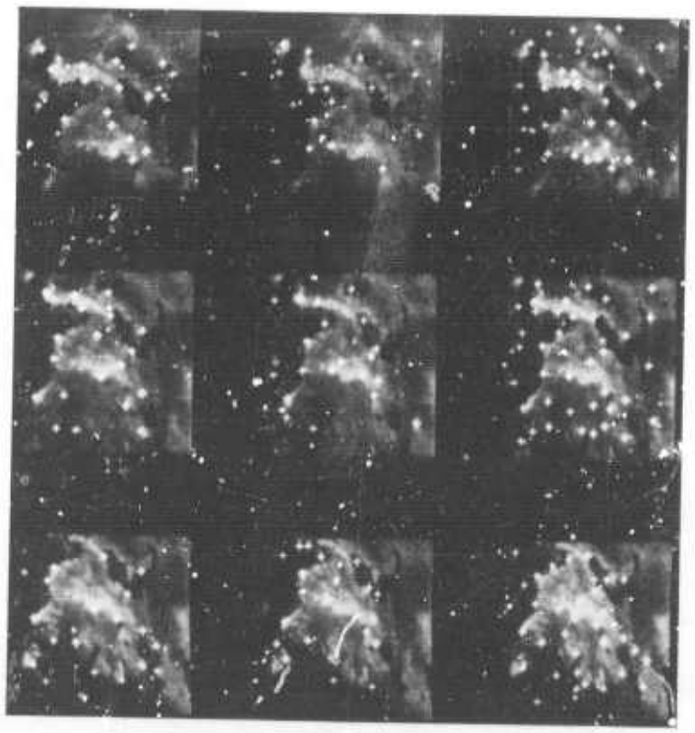


Figure 4 Three Measures of the Interest of a Point.

The first column of images shows peaks in ordinary variance for three images from the Night Vision Laboratory's terrain model. The second column shows peaks in directed variance for the same three images. The third column shows peaks in edged variance.

a) A hierarchy of images, each the 3 × 3 reduction of its parent.

Figure 5 Reduction Matching

b) Matching points found by expanding a grid through the reduction image hierarchy.

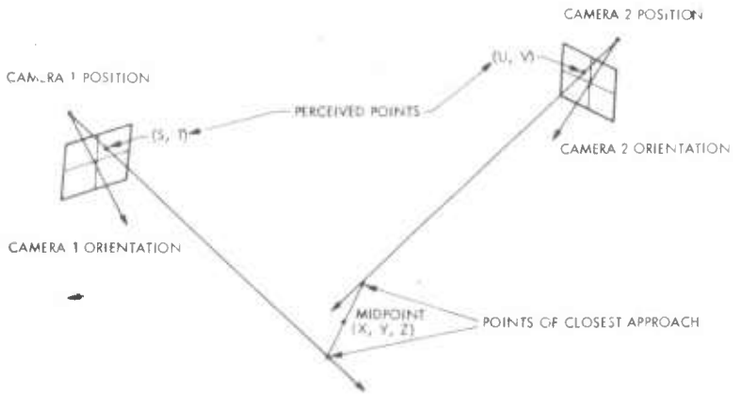


Figure 6 Point Position Calculation

The points (S,T) and (U,V) are projected through their respective cameras. Their intersection (X,Y,Z) is defined to be the mid-point between the points of closest approach for these rays.

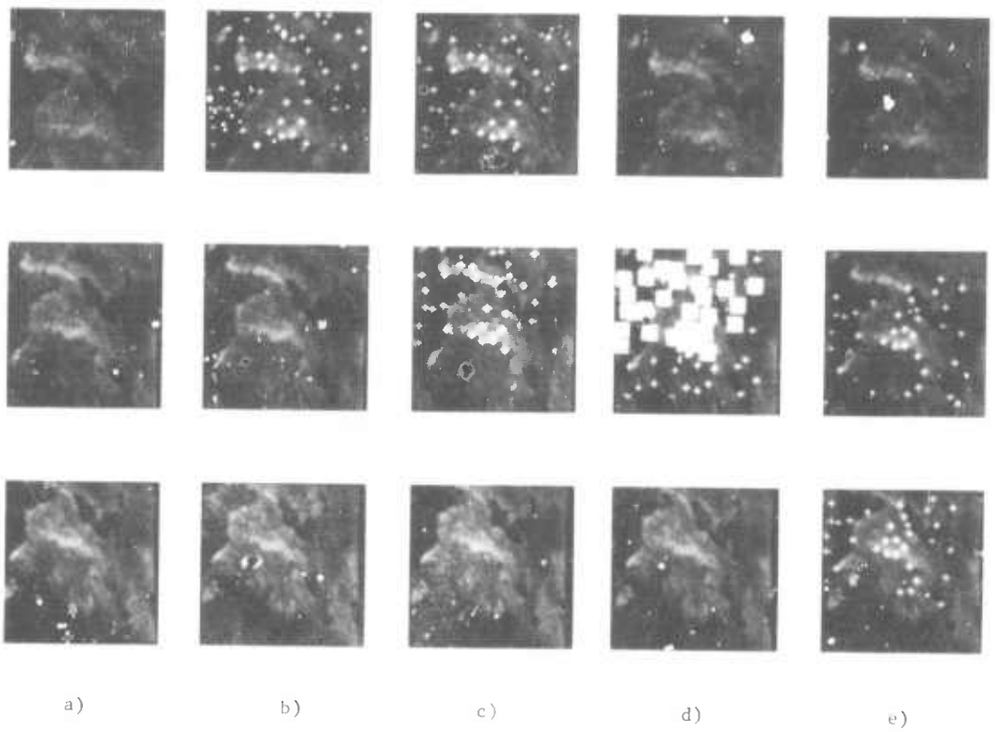


Figure 7 An Example of Control-Point Handling for Bootstrap Stereo

a) The original sequence of 3 images.
b) The interesting points in Image 1.
c) The matched points between Images 1 and 2.
d) The areas of Image 2 covered by matches, with interesting points found in the uncovered areas.
e) The control points in Image 2 matched to image 3.

STATUS REPORT ON THE ADVANCED FLEXIBLE PROCESSOR

BY
G.R. Allen

Information Sciences Division
Control Data Corporation

A major milestone was reached in early February in the development of the Advanced Flexible Processor (AFP). Construction of the first AFP was completed and checkout was initiated on the AFP and interface to the system controller (PDP 11/70). All of the people who labored for several years to design, simulate, and fabricate the AFP were extremely pleased to have reached this milestone. An appropriate celebration was held to mark the occasion. More good news - testing and debugging the machine has gone extremely well.

The design has now been fully verified. No major design errors have been uncovered in the checkout process (6-7 days per week, 3 shifts per day) as of April 2, 1980. All available diagnostics (over 2000 lines of code have been run successfully with the hardware passing old tests. The first set of user application code has been run and is now nearly operational.

Progress in bringing up the AFP has been phenomenal and can be attributed to three factors. The machine was fully simulated at the gate level including gate and wire delays. Proven technology was used in the construction including power distribution, freon cooling, printed circuit board construction, and integrated circuits. An instruction-level simulator was developed to model the hardware which eliminated the problem of debugging untested programs on untested hardware.

Construction of the second unit (to be delivered to Carnegie-Mellon University) is proceeding in parallel with the checkout effort; however, this is taking second place to getting the first machine working. The cabinet construction is complete and the cooling system is starting into final test. With few exceptions, the printed circuit boards and integrated circuits for the CMU machine are on hand. Following completion of major testing on the first AFP, changes will be installed in the CMU unit and checkout will be initiated.

Work has also been started on a nine-processor system to be completed in mid-1981. This system will provide a computer capability of over 2 billion arithmetic operations per second.